FIT5037: Network Security
Asymmetric key cryptography

Faculty of Information Technology
Monash University

March 19, 2019

Copyright Notice

Lecture 2: Asymmetric key cryptography

Lecture Topics:

- Symmetric key cryptography
- **Asymmetric key cryptography**
- Pseudorandom Number Generators and hash functions
- Authentication Methods and AAA protocols
- Security at Network layer
- Security at Network layer (continued)
- Security at Transport layer
- Security at Application layer
- Computer system security and malicious code
- Computer system vulnerabilities and penetration testing
- Intrusion detection
- Denial of Service Attacks and Countermeasures / Revision

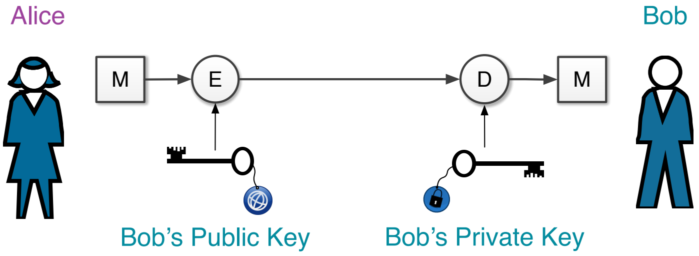
Outline

- Asymmetric Cryptography
- RSA
- DH Key Exchange
- DSA
- Elliptic Curve

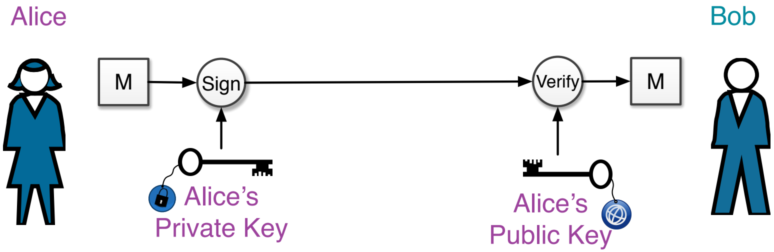
Asymmetric Cryptography

- Two keys are used: Public and Private
- Two entities in a communication: Sender and Receiver
- Two different applications depending on which key of which participant is used:
    - Confidentiality
    - Authenticity, Integrity and Non-repudiation
- Asymmetric because
    - those who encrypt the message cannot decrypt it
    - those who verify the signature cannot create it

Asymmetric Cryptography



Public Key Cryptography: Confidentiality



Public Key Cryptography: Authenticity and Non-repudiation

Why asymmetric encryption?

- Developed to address two important issues:
    - key distribution: how to have secure communications in general without having to trust a KDC with your key
    - digital signatures: how to verify a message comes intact from the claimed sender
- public invention due to Whitfield Diffie and Martin Hellman at Stanford University in 1976
    - known earlier in classified community

Symmetric vs. Public Key Cryptography

Symmetric Key Cryptography

Advantages:

- **Faster:** uses fast round functions (e.g. XOR, shift and table lookups)
- **Smaller key sizes (same computational security):** no mathematical attack

Disadvantages:

- **Key exchange problem:** both parties require the same key to establish a secure connection
- **Key management problem:** requires $\frac{n(n-1)}{2}$ unique keys for **n** users

Symmetric vs. Public Key Cryptography

Public Key Cryptography

Advantages:

- **Key exchange is easy:** public keys can be sent over unsecured channel
- **Key management is easier:** requires $2n$ keys for **n** users

Disadvantages:

- **Slower:** are based on mathematically hard problems that require computationally expensive operations (e.g. modular exponentiation)
- **Larger key sizes (same level of computational security):** due to more efficient mathematical attacks (compared to brute force)

Public-key Requirements

1. It must be computationally easy for any entity to generate a public and private key pair
2. It must be computationally easy for any entity to encrypt a message using the public key of a recipient to achieve confidentiality
3. It must be computationally easy for the recipient of a confidential message to decrypt using her private key
4. It must be computationally infeasible for an adversary to recover a message knowing the public key and the ciphertext
5. It must be computationally infeasible for an adversary to recover the private key knowing the public key

Public Key Application

Three categories of use:

- **Confidentiality:** Sender uses the public key of the receiver to encrypt a message
- **Authentication and Non-repudiation:** Sender uses her private key to generate a digital signature for a message
- **Key exchange:** Sender and Receiver use a public key method specifically used for key exchange or use the confidentiality feature (public key of the receiver is used to encrypt the symmetric key)

Some algorithms are suitable for all uses, others are specific to one

Asymmetric Encryption Algorithms

- Similar to symmetric encryption brute force/exhaustive search attack is always possible
- But keys used are large (e.g. 1024 bits in RSA) due to existence of more efficient mathematical attacks
- Security relies on difficulty of a mathematical problem
- Requires the use of very large numbers
- Hence is slow compared to symmetric encryption

Asymmetric Encryption Algorithms

Two widely used algorithms:

- RSA Algorithm
  - developed by Ron Rivest, Adi Shamir and Len Adleman at MIT proposed in 1978
  - security relies on the cost of factoring numbers with two large prime factors
- Diffie-Hellman Algorithm
  - developed by Whitfield Diffie and Martin Hellman in 1976
  - security relies on the difficulty of computing discrete logarithms

RSA Algorithm: Key Generation

To generate RSA key pairs with N-bit security:

1. Choose two random prime numbers p and q of size $\frac{N}{2}$ bits
2. Calculate $n = pq$ and $\phi(n) = (p-1)(q-1)$
3. Choose e and check $gcd(e, \phi(n)) = 1$
4. Calculate d such that $ed \bmod \phi(n) = 1$ (extended Euclidean algorithm)
5. Publish e, n as public key

- e is the public exponent (65537 is a common choice in practice)
- d is the private exponent

RSA Algorithm: Confidentiality

- Encryption (confidentiality, public key of receiver)
  - ciphertext: $C = M^e \bmod n$
- Decryption (private key of receiver)
  - recover message: $M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$
- Euler's Theorem: if $a \in \mathbb{Z}^*_n$ then $a^{\phi(n)} \bmod n \equiv 1 \pmod{n}$
- Since $ed \bmod \phi(n) = 1$ then $ed = k \cdot \phi(n) + 1$ for some k (division algorithm):
  - $M^{ed} \bmod n = M^{k \cdot \phi(n)+1} \bmod n =$
  - $M \cdot M^{k \cdot \phi(n)} \bmod n =$
  - $M \cdot (M^{\phi(n)} \bmod n)^k \bmod n =$ (properties of modular arithmetic)
  - $M \cdot 1^k \bmod n = M$ (Euler's theorem)

RSA Algorithm: Long Documents

- RSA can be used similar to a block cipher
  - each block must have a binary value $M < n$
  - in practice block size is k bits, $2k < n < 2k + 1$
- This however will be very inefficient
- In practice public key is used for authentication and encryption of session keys
- Session keys are then used to encrypt long documents using symmetric key encryption

Requirements of RSA (Confidentiality)

For RSA algorithm to work satisfactorily, the following requirements must be met:

1. computationally easy to generate key pairs: e, d, and n (refer to key generation steps)
2. computationally easy to encrypt $C = M^e \bmod n$
3. computationally easy to decrypt $M = C^d \bmod n$
4. computationally infeasible to recover M knowing only e and n
5. computationally infeasible to recover d from e and n

Diffie-Hellman Key Exchange

- First public-key algorithm (publicly known)
- proposed by Diffie and Hellman in 1976 along with the exposition of public key concepts
  - note: now known that James Ellis (UK CESG) secretly proposed the concept in 1970
- A practical method for secure exchange of a secret key
- Used in a number of protocols

Diffie-Hellman Key Exchange

- a public-key algorithm used to distribute symmetric keys
    - cannot be used to exchange an arbitrary message
    - rather it can establish a common key
    - known only to the two participants
- value of key depends on the participants (and their private and public key information)
- based on exponentiation in a finite (Galois) field (modulo a prime or a polynomial), easy
- security relies on the difficulty of computing discrete logarithms (similar to factoring), hard

Diffie-Hellman Key Exchange

1. Alice and Bob agree on global parameters:
    - p, a large prime integer
    - g, a generator/primitive root of p (of $\mathbb{Z}^*{}_{\mathsf{p}}$)
2. Alice and Bob, each:
    - chooses a private key (number)
        - Alice: $X_A < p$
        - Bob: $X_B < p$
    - compute their public key
        - Alice: $Y_A = g^{X_A} \bmod p$
        - Bob: $Y_B = g^{X_B} \bmod p$
3. Alice and Bob exchange their public key Y_A and Y_B
4. Alice and Bob calculate the shared secret:
    - Alice: $K_{AB} = Y_B^{X_A} \bmod p$
    - Bob: $K_{BA} = Y_A^{X_B} \bmod p$

- K_{AB} is used as session key in symmetric key encryption scheme between Alice and Bob

Diffie-Hellman Key Exchange: Shared Key

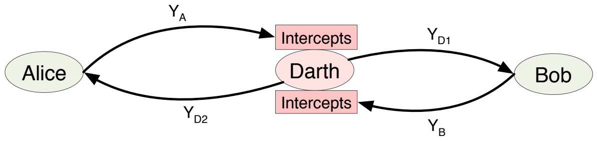
Shared session key for Alice and Bob is the same:

- $K_{AB} = Y_B^{X_A} \bmod p =$
- $(g^{X_B} \bmod p)^{X_A} \bmod p =$
- $g^{X_B \cdot X_A} \bmod p =$
- $g^{X_A \cdot X_B} \bmod p =$
- $(g^{X_A} \bmod p)^{X_B} \bmod p =$
- $Y_A^{X_B} \bmod p = K_{BA}$
- attacker needs X_A or X_B, or must solve discrete log (from Y_A or Y_B, and public parameters p and g) to recover the session key

Key Exchange Protocols

- users could create random private/public DH keys each time they communicate
- users could create a known private/public DH key and publish in a directory, then consulted and used to securely communicate with them
- both of these are vulnerable to a Man-in-the-Middle Attack
- All public key algorithms are vulnerable to Man-in-the-Middle Attack unless public keys can be authenticated
- Hence; authentication of the public keys is needed

Man-in-the-Middle Attack



- Darth creates two sets of private / public key pairs
- Alice transmits her public key to Bob
- Darth intercepts this and transmits his first public key to Bob. Darth also calculates a shared key with Alice
- Bob receives the public key and calculates the shared key (with Darth instead of Alice)
- Bob transmits his public key to Alice
- Darth intercepts this and transmits his second public key to Alice. Darth calculates a shared key with Bob
- Alice receives the key and calculates the shared key (with Darth instead of Bob)
- Darth can then intercept, decrypt, re-encrypt, forward all essages between Alice and Bob

How to Prevent MitM

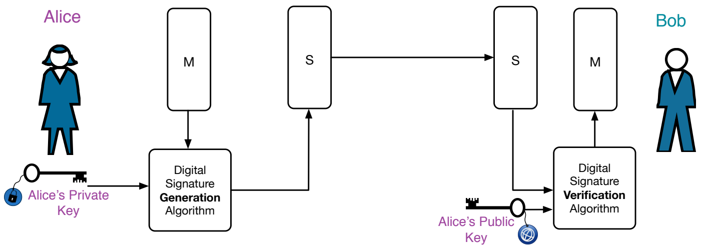
- Public keys must be authenticated
- Physical exchange of the public keys (web of trust in PGP/GPG)
- Use of Certificate Authorities (trusted third parties)
    - Certificate Authorities verify identity of participants
    - then digitally sign public key of participants (create the certificate)

Digital Signature

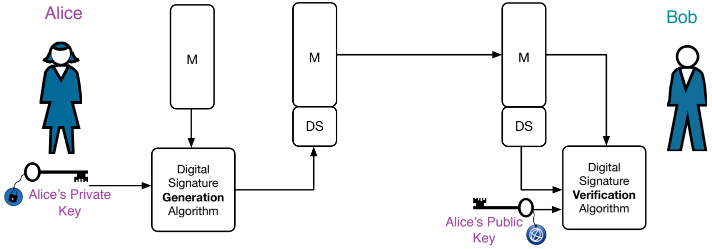
- Digital signatures reproduce the electronic version of the normal signatures
    - proof of identity (authenticity of the message origin)
    - proof of message integrity
    - non-repudiation
- Differences with MAC
    - provides authenticity of origin down to exactly one entity
        - MAC does not distinguish between two participants
    - provides non-repudiation
        - MAC does not distinguish between two participants

Digital Signature Construction

Digital Signature with Message Recovery

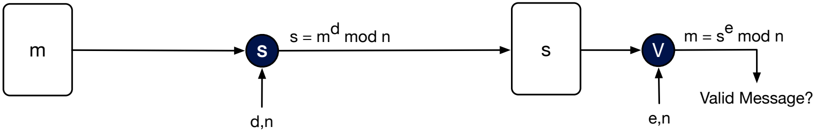

Alice
M
S
S
M
Bob
Digital Signature Generation Algorithm
Alice's Private Key
Digital Signature Verification Algorithm
Alice's Public Key

Digital Signature without Message Recovery


Alice
M
M
DS
M
DS
Bob
Digital Signature Generation Algorithm
Alice's Private Key
Digital Signature Verification Algorithm
Alice's Public Key

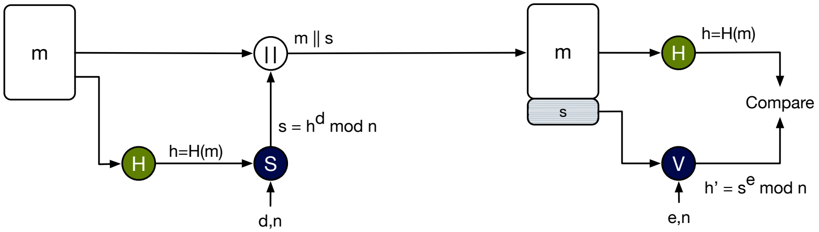
RSA Signature with Message Recovery

- what if $m > n$? break down the message to blocks $m_1, m_2, \ldots, m_t$ where $\forall i, m_i < n$
- requires a way to verify $m \in \mathcal{M}$ where $\mathcal{M}$ is the set of all possible messages
- less efficient than the method without message recovery
- is not used in practice
- vulnerable to existential forgery (in original form)
    - feature of modular arithmetic (how?)



RSA signature without message recovery

- more efficient as hash functions are faster than modular exponentiation
- can make sure message digest is always less than modulus
- forgery requires a collision on the hash function



Digital Signature Algorithm (DSA) [1]

- Standardised in 1991 by NIST: Digital Signature Standard (DSS)
- requires a hash function $h : \{0,1\}^* \to \mathbb{Z}_q$ for some integer q
- Key generation algorithm is as follows for SHA1 (with 160 bit digest):
  1. Select a prime number q: $2^{159} < q < 2^{160}$
  2. Choose t: $0 \leq t \leq 8$ and prime p: $2^{511+64t} < p < 2^{512+64t}$ such that $q | (p-1)$
  3. Select a generator g of the unique cyclic group of order q in $\mathbb{Z}^*{}_p$ as follows:
     - Select an element $g_1 \in \mathbb{Z}^*{}_p$ and compute $g = {g_1}^{(p-1)/q} \bmod p$
     - If $g = 1$ then go to previous step
  4. Select a random integer a: $1 \leq a \leq q-1$
  5. Compute $y = g^a \bmod p$
  6. A's public key is (p, q, g, y); A's private key is a

[1] Handbook of Applied Cryptography: Chapter 11

DSA

- Why q is chosen first and then p is searched for?
- How step 2 is done (algorithm)?
- Can two users share p, q, and g (is it secure)?

DSA Signature Generation [2]

- To generate a signature for the message m with A's private key a and public parameters p, q, g:

1. Select a random secret integer k: $0 < k < q$
2. Compute $r = (g^k \bmod p) \bmod q$
3. Compute $k^{-1} \bmod q$
4. Compute $s = k^{-1}(h(m) + ar) \bmod q$
5. A's signature for m is the pair (r, s)

[2] Handbook of Applied Cryptography: Chapter 11

DSA Signature Verification [3]

To verify a signature (r, s) for the message m with A's public key p, q, g, y:

1. Verify $0 < r < q$ and $0 < s < q$ otherwise reject
2. Compute $w = s^{-1} \bmod q$
3. Compute $h(m)$
4. Compute $u_1 = w \cdot h(m) \bmod q$
5. Compute $u_2 = rw \bmod q$
6. Compute $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$
7. if $v = r$ accept otherwise reject

Work out why this verifies the signature

[3] Handbook of Applied Cryptography: Chapter 11

Security of DSA

- Relies on difficulty of discrete logarithm
  - difficulty of DLP in $\mathbb{Z}^*{}_p$
  - difficulty of DLP in cyclic subgroup of order q
- Latest NIST document: FIPS 186-4
- Recommended parameter sizes for $2^{L-1} < p < 2^L$ and $2^{N-1} < q < 2^N$:
  - L=1024, N=160
  - L=2048, N=224
  - L=2048, N=256
  - L=3072, N=256
- DSA is a variation of ElGamal signature scheme

Elliptic Curve Cryptography [4]

General Definition: An elliptic curve E over a field $K = F_{p^m}$ is defined by an equation:

1. $E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ where:
   - $a_1, a_2, a_3, a_4, a_6 \in F_{p^m}$
   - Δ is the discriminant of E and $\Delta \neq 0$
   - $\Delta = -{d_2}^2 d_8 - 8{d_4}^3 - 27{d_6}^2 + 9d_2d_4d_6$
     - $d_2 = {a_1}^2 + 4a_2$
     - $d_4 = 2a_4 + a_1a_3$
     - $d_6 = {a_3}^2 + 4a_6$
     - $d_8 = {a_1}^2 a_6 + 4a_2a_6 - a_1a_3a_4 + a_2{a_3}^2 - a_4^2$

[4] Guide to Elliptic Curve Cryptography - D. Hankerson, A. Menezes, S. Vanstone: Chapter 3

ECC[5]

- The equation (1) is called a *Weierstrass equation*
- E is defined over K since coefficients $(a_1, a_2, a_3, a_4, a_6)$ are elements of of K (sometimes expressed as E/K or $E(K)$)
- the requirement $\Delta \neq 0$ is to make sure the curve is *smooth* (there are no points that has more than one distinct tangent)

[5]Guide to Elliptic Curve Cryptography - D. Hankerson, A. Menezes, S. Vanstone: Chapter 3

Simplified Weierstrass Equations[6]

- Two elliptic curves are called isomorphic if one can be transformed to the other through *admissible change of variables*
- A Weierstrass equation (1) can be transformed to:

(2) $y^2 = x^3 + ax + b$ where:

- $a, b \in K$
- $\Delta = -16(4a^3 + 27b^2)$

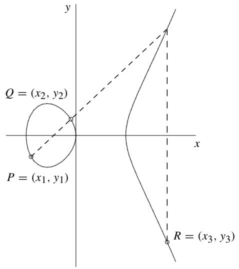
[6]Guide to Elliptic Curve Cryptography - D. Hankerson, A. Menezes, S. Vanstone: Chapter 3

ECC Remarks

- Protocols that we study in this subject only use two underlying fields:
  - F_p, p large prime ($>$192 bits) (characteristic p)
  - F_{2^m} and simplified equation $y^2 + xy = x^3 + ax^2 + b$ (characteristic 2)
  - of these two we only look at the basics of the first one
- For the rest of the lecture notes unless otherwise specified $E/K \equiv E/F_p$ (although the mathematics may be applicable to F_{p^m} $p > 3$)

ECC Primitive Operation: Point Addition

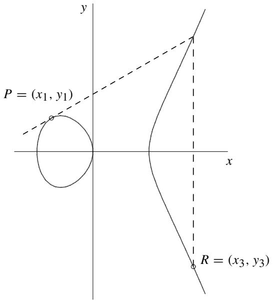
- For the elliptic curve over finite field $E(K) : y^2 = x^3 + ax + b$, let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two distinct points on the curve then the *sum* $R = (x_3, y_3)$ of the points P and Q, $R = P + Q$, is defined as:
  1. draw a line through P and Q
  2. the line intersects the curve at a third point (feature of elliptic curve)
  3. R is the reflection of the third point about the x-axis



- $x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2$
- $y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1$

ECC Primitive Operation: Point Doubling

- The *double* R of P, $R = 2P$ (also expressed as $[2]P$) is defined as:
  1. draw the tangent line to the elliptic curve at point P
  2. the line intersects the elliptic curve at a second point
  3. R is the reflection of the second point about the *x*-axis



- $x_3 = \left(\frac{3{x_1}^2 + a}{2y_1}\right)^2 - 2x_1$
- $y_3 = \left(\frac{3{x_1}^2 + a}{2y_1}\right)(x_1 - x_3) - y_1$

ECC Identity Element

For an elliptic curve E: $y^2 = x^3 + ax + b$ over a field K with $char(K) > 3$ a special point called the *point at infinity* expressed as O (or ∞) is defined such that:

- $\forall P \in E(K), P + O = O + P = P$
  - similar to zero for addition
  - think of this as a vertical line that intersects the curve at infinity
- to follow the geometric demonstration logic:
  1. draw a line that passes through P and O, the line will be vertical
  2. the line will intersect the curve at a third point (reflection of P about the x-axis)
  3. the reflection of that point is P itself

ECC Point Inverses

For an elliptic curve E: $y^2 = x^3 + ax + b$ over a field K with $char(K) > 3$

- Inverse element of point addition operation:
  - If $P = (x, y) \in E(K)$ then $-P = (x, -y)$
  - $P - P = O$
  - $-O = O$
- following the geometric definition:
  - $P + (-P) = O$ hence the line must pass through P and O
  - the line will be vertical
  - the only other intersection between the vertical line and the curve besides P and O is the reflection of P about the x-axis

ECC Group Law

An elliptic curve E: $y^2 = x^3 + ax + b$ over a prime field K with $char(K) > 3$ with point addition and point doubling operation and identity element O forms an Abelian group.

- The set $E(K)$ with binary operation $+$ (point addition) forms an Abelian group:
  1. associative: $\forall P, Q, R \in E(K) : P + (Q + R) = (P + Q) + R$
  2. identity element O
  3. inverse element $\forall P \in E(K), \exists \; -P \in E(K)$
  4. commutative: $\forall P, Q \in E(K), P + Q = Q + P \in E(K)$
- A point $P = (x, y)$ is said to be on curve E over field K if $x, y \in K$ and (x, y) satisfies the curve equation
- Note: Geometric representation is used to demonstrate the point addition and point doubling operations and to derive the formulae, calculations are done in underlying finite field

Example: ECC over prime field $\mathbb{F}_{29}$

- Let $p = 29$, $a = 4$, and $b = 20$, then the elliptic curve $E : y^2 = x^3 + 4x + 20$ is defined over $\mathbb{F}_{29}$.
    - $\Delta = -16(4a^3 + 27b^2) = -176896 \not\equiv 0 (\bmod 29)$
    - The set of points:

∞	$(2,6)$	$(4,19)$	$(8,10)$	$(13,23)$	$(16,2)$	$(19,16)$	$(27,2)$
$(0,7)$	$(2,23)$	$(5,7)$	$(8,19)$	$(14,6)$	$(16,27)$	$(20,3)$	$(27,27)$
$(0,22)$	$(3,1)$	$(5,22)$	$(10,4)$	$(14,23)$	$(17,10)$	$(20,26)$	
$(1,5)$	$(3,28)$	$(6,12)$	$(10,25)$	$(15,2)$	$(17,19)$	$(24,7)$	
$(1,24)$	$(4,10)$	$(6,17)$	$(13,6)$	$(15,27)$	$(19,13)$	$(24,22)$	

How do we check if a point is on the curve? let's say (4,19)?

ECC: $\mathbb{F}_{29}$ Point Addition Toy Example

- Let $P = (5, 22)$ and $Q = (16, 27)$ from the set of the points (previous slide), to calculate $R = (x_3, y_3)$:
  - $x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \bmod 29$
    $x_3 = \left((27 - 22) \times (16 - 5)^{-1}\right)^2 - 5 - 16 \bmod 29$
    $x_3 = (5 \times 11^{-1})^2 - 21 \bmod 29$, $11^{-1} \bmod 29 = 8$
    $x_3 = 40^2 + 8 \bmod 29 = 13$
  - $y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1$
    $y_3 = 40 \times (5 - 13) - 22 \bmod 29$
    $y_3 = 40 \times 21 + 7 \bmod 29 = 6$
  - $R = (13, 6) \in E(F_{29})$

ECC: $\mathbb{F}_{29}$ Point Doubling Toy Example

- Let $P = (5, 22)$ then $2P = (x_3, y_3)$:
  - $x_3 = \Big(\dfrac{3{x_1}^2 + a}{2y_1}\Big)^2 - 2x_1$
    $x_3 = \big((3 \times 5^2 + 4) \times (2 \times 22)^{-1}\big)^2 - 2 \times 5 \bmod 29$
    $x_3 = (79 \times 44^{-1})^2 - 10 \bmod 29,\ 44^{-1} \bmod 29 = 2$
    $x_3 = (21 \times 2)^2 + 19 \bmod 29$
    $x_3 = 42^2 + 19 \bmod 29 = 1783 \bmod 29 = 14$
  - $y_3 = \Big(\dfrac{3{x_1}^2 + a}{2y_1}\Big)(x_1 - x_3) - y_1$
    $y_3 = 42 \times (5 - 14) - 22 \bmod 29$
    $y_3 = 42 \times (-9) + 7 \bmod 29$
    $y_3 = 42 \times 20 + 7 \bmod 29 = 847 \bmod 29 = 6$
  - $R = (14, 6) \in E(F_{29})$

ECC: Point at infinity and Inverse elements Toy Example

- From the set of points observe the points:
  - $P = (2, 6)$ and $Q = (2, 23)$
    - $Q = -P$ as $-P = (2, -6 \bmod 29) = (2, 23)$
    - To calculate $R = P + Q = (x_3, y_3)$

      $x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2$ where $x_2 - x_1 = 0$ and we will have division by zero hence point at infinity and $P + Q = O$
  - $P = (1, 5)$ and $Q = (1, 24)$
  - $P = (15, 2)$ and $Q = (15, 27)$

ECC: Point multiplication by scalar

- Point multiplication by scalar α is defined as adding a point α times to itself
  - $Q = \alpha P = \overbrace{P + P + \cdots + P}^{\alpha \text{ times}}$
  - This operation is easy $P \rightarrow \alpha P$
    - using repeated point doubling and point addition (at most $2log_2\alpha$ of group operation)
  - The inverse of this operation however is *believed* to be hard
    - knowing the points P and $Q = \alpha P$ it is computationally hard to find α
    - this is defined as the Elliptic Curve Discrete Logarithm Problem (ECDLP)

ECDH Key Exchange

1. Alice and Bob agree on a curve and P a generator (base) point on curve
   - Alice chooses a random scalar private key α and calculates a public key $A = \alpha P$ (point on curve)
   - Bob chooses a random scalar private key β and calculates a public key $B = \beta P$ (point on curve)
2. Alice and Bob exchange their public keys
   - Alice calculates the shared key $K_{ab} = \alpha B$ (point on curve)
   - Bob calculates the shared key $K_{ba} = \beta A$ (point on curve)

- Shared key is the same at both sides:
  - $K_{ab} = \alpha B = \alpha(\beta P) = \alpha\beta P$
  - $K_{ba} = \beta A = \beta(\alpha P) = \beta\alpha P$

ECC Security

- In measuring the security of any discrete logarithm knowing the group order is required
- Let E be an elliptic curve defined for F_{p^m}, the number of points in $E(F_{p^m})$ denoted $\#E(F_{p^m})$ (or $|E(F_{p^m}))|$) is called the *order* of E over F_{p^m}
- Hasse's Theorem: for $E(F_{p^m})$: $p^m + 1 - 2\sqrt{p^m} \le \#E(F_{p^m}) \le p^m + 1 + 2\sqrt{p^m}$
- There is an efficient algorithm by Schoof to calculate the number of points on $E(F_{p^m})$
- For a point $P \in E(F_{p^m})$ the *order* is defined as the smallest integer l such that $lP = O$ (the identity element)
    - the ECDLP is as hard as $\sqrt{q}$ where q prime and $q = l$ or $q|l$ so a point of sufficiently large q is desirable (e.g. $q > 160$)
    - this will create a cyclic (or near cyclic) subgroup

Common Curves: NIST P256

- NIST P256
  - Published by NIST in FIPS 186-4
  - curve: $y^2 = x^3 - 3x + b (\mathrm{mod} p)$
  - $p = 2^{256} - 2^{224} + 2^{192} + 2^{9}6 - 1$

```
b = 5ac635d8 aa3a93e7 b3ebbd55 769886bc
    651d06b0 cc53b0f6 3bce3c3e 27d2604b
```

  - base point:

```
x = 6b17d1f2 e12c4247 f8bce6e5 63a440f2
    77037d81 2deb33a0 f4a13945 d898c296


y = 4fe342e2 fe1a7f9b 8ee7eb4a 7c0f9e16
    2bce3357 6b315ece cbb64068 37bf51f5
```

  - n the order of the base point:

```
n = 1157920892103562487626974469494075735299
    969552241357603424222590610685120443369
```

- implementation of this curve is mandatory in TLS 1.3

Common Curves: Curve 25519

- Published by D. J. Bernstein in 2005[7]
  - curve: $By^2 = x^3 + Ax^2 + x$ where $A, B \in F_p$ and $p > 3$
    - expressed as a Montgomery curve: $y^2 = x^3 + 48662x^2 + x$
  - $p = 2^{255} - 19$
  - base point is $x = 9$ (compressed elliptic point encoding)
  - does not require point validation
  - has a fast multiplication algorithm to compute αP from $P \in E(F_p)$ and $\alpha \in F_p$

[7] Curve25519: new Diffie-Hellman speed records

References

- Handbook of Applied Cryptography: Chapter 11
- FIPS PUB 186-4: Digital Signature Standard
- Guide to Elliptic Curve Cryptography - D. Hankerson, A. Menezes, S. Vanstone: Chapter 3
- Curve25519: new Diffie-Hellman speed records