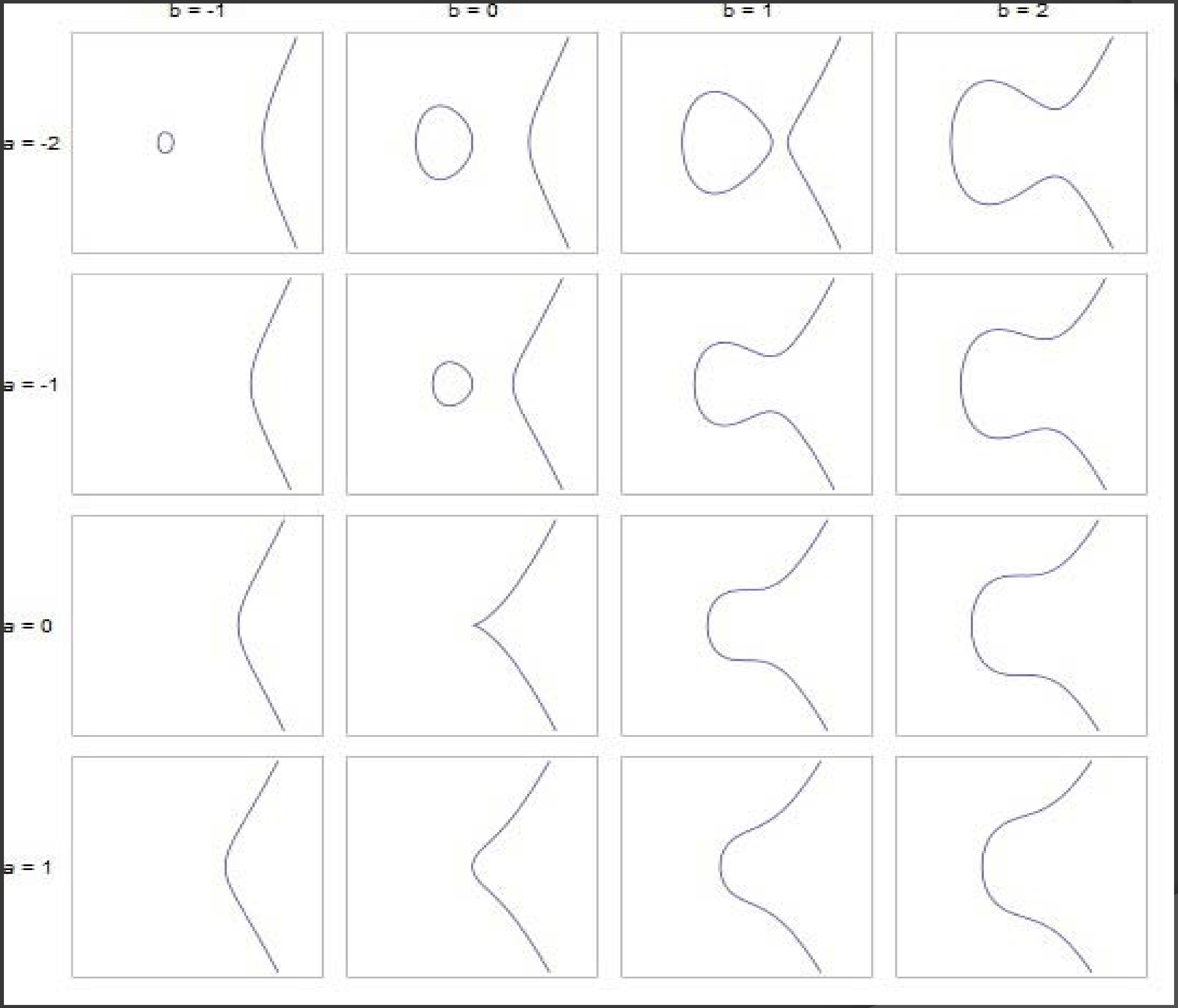
ELLIPTIC CURVE CRYPTOGRAPHY

By

Abhijith Chandrashekar
and
Dushyant Maheshwary

Introduction

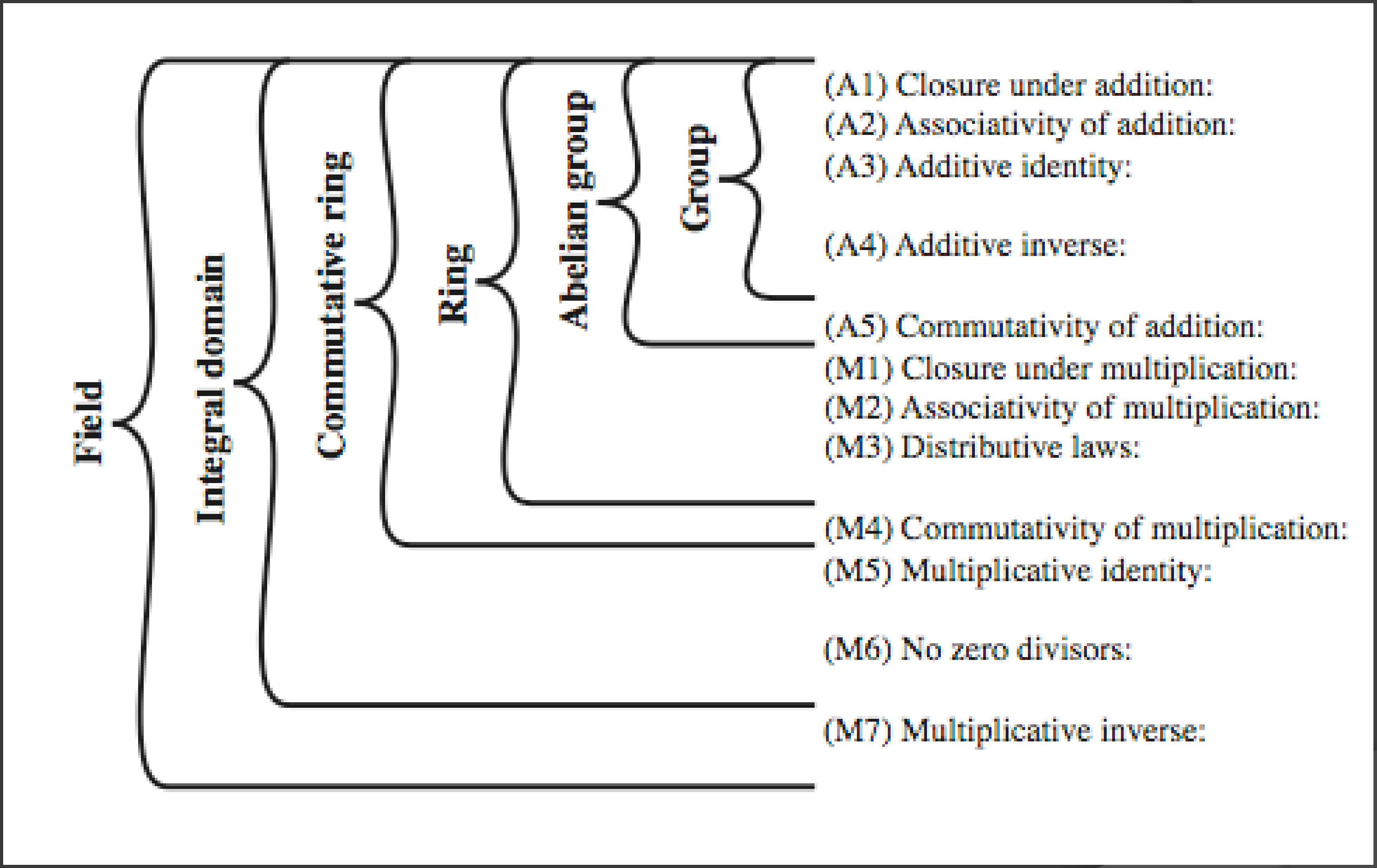
- What are Elliptic Curves?
  - Curve with standard form $y^2 = x^3 + ax + b$ $a, b \in \mathbb{R}$
- Characteristics of Elliptic Curve
  - Forms an abelian group
  - Symmetric about the x-axis
  - *Point at Infinity* acting as the identity element



Examples of Elliptic Curves

Finite Fields

- aka Galois Field
- $GF(p^n)$ = a set of integers $\{0, 1, 2, \ldots, p^n - 1\}$
  where p is a prime, n is a positive integer
- It is denoted by $\{F, +, x\}$
  where + and x are the group operators



Group, Ring, Field

Why Elliptic Curve Cryptography?

- Shorter Key Length
- Lesser Computational Complexity
- Low Power Requirement
- More Secure

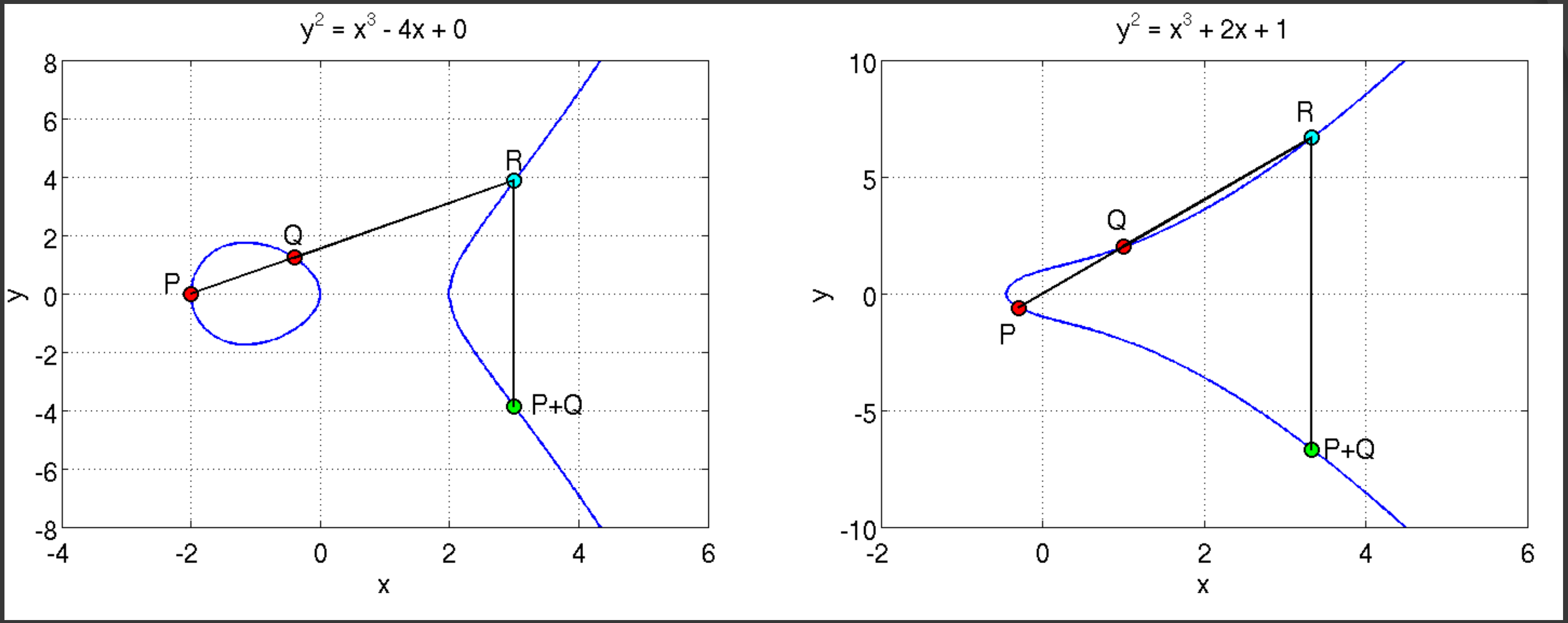
Comparable Key Sizes for Equivalent Security

Symmetric Encryption (Key Size in bits)	RSA and Diffie-Hellman (modulus size in bits)	ECC Key Size in bits
56	512	112
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

What is Elliptic Curve Cryptography?

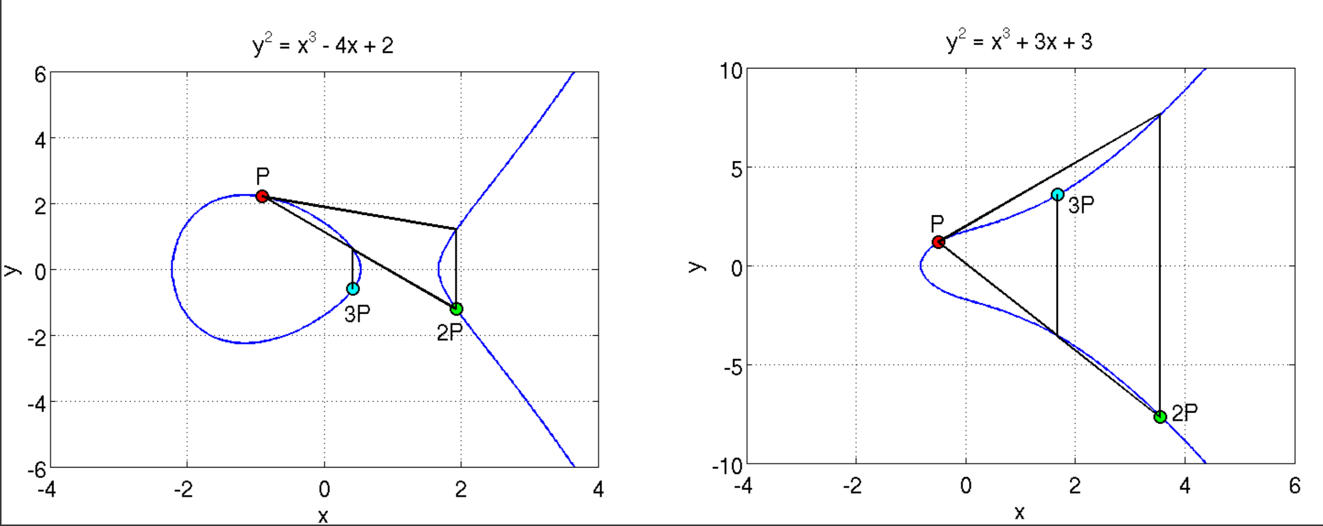
- Implementing Group Operations
  - Main operations - point addition and point multiplication
  - Adding two points that lie on an Elliptic Curve – results in a third point on the curve
  - Point multiplication is repeated addition
  - If P is a known point on the curve (aka Base point; part of domain parameters) and it is multiplied by a scalar k, Q=kP is the operation of adding P + P + P + P… +P (k times)
  - Q is the resulting public key and k is the private key in the public-private key pair

What is Elliptic Curve Cryptography?



- Adding two points on the curve
- P and Q are added to obtain P+Q which is a reflection of R along the X axis

What is Elliptic Curve Cryptography?



- A tangent at P is extended to cut the curve at a point; its reflection is 2P
- Adding P and 2P gives 3P
- Similarly, such operations can be performed as many times as desired to obtain Q = kP

What is Elliptic Curve Cryptography?

- Discrete Log Problem
  - The security of ECC is due the intractability or difficulty of solving the inverse operation of finding k given Q and P
  - This is termed as the discrete log problem
  - Methods to solve include brute force and Pollard's Rho attack both of which are computationally expensive or unfeasible
  - The version applicable in ECC is called the Elliptic Curve Discrete Log Problem
  - Exponential running time

ECC in Windows DRM v2.0

A Practical Example :

Finite field chosen

p = 785963102379428822376694789446897396207498568951

Gx = 771507216262649826170648268565579889907769254176

Gy = 390157510246556628525279459266514995562533196655

$$y^2 = x^3 + 317689081251325503476317476413827693272746955927x + 790528966078787587181205720257185354321100651934$$

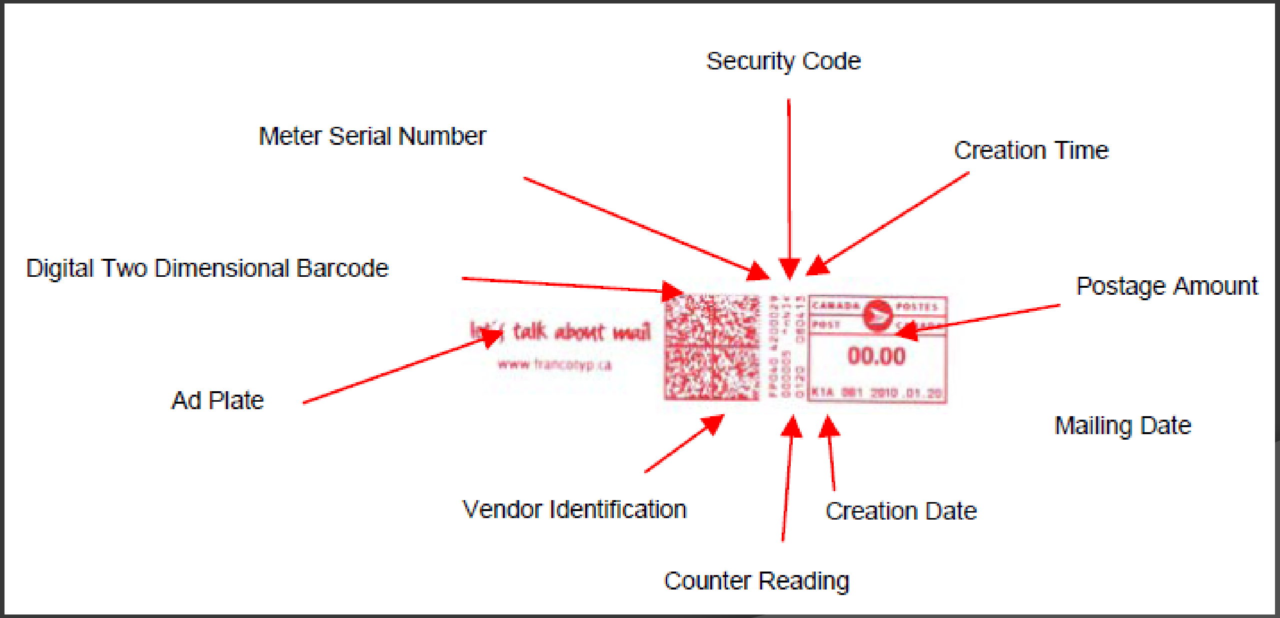
Gx and Gy constitute the agreed upon base point (P) and the numbers in the above equation are values for the parameters a and b

Elliptic Curve Schemes

- Elliptic Curve Digital Signature Algorithm (ECDSA)
- Elliptic Curve Pintsov Vanstone Signature(ECPVS)
- Elliptic Curve Diffie-Hellman (ECDH)

Elliptic Curve Digital Signature Algorithm (ECDSA)

- Elliptic curve variant of Digital Signature Algorithm



Canadian postage stamp that uses ECDSA

ECDSA

- Signature Generation

Once we have the domain parameters and have decided on the keys to be used, the signature is generated by the following steps.

1. A random number k, $1 \leq k \leq n-1$ is chosen
2. $kG = (x_1,y_1)$ is computed. x_1 is converted to its corresponding integer x_1'
3. Next, $r = x_1 \bmod n$ is computed
4. We then compute $k^{-1} \bmod q$
5. $e = HASH(m)$ where m is the message to be signed
6. $s = k^{-1}(e + dr) \bmod n$.

We have the signature as (r,s)

ECDSA

- Signature Verification

At the receiver's end the signature is verified as follows:

1. Verify whether r and s belong to the interval [1, n-1] for the signature to be valid.
2. Compute e = HASH(m). The hash function should be the same as the one used for signature generation.
3. Compute $w = s^{-1} \bmod n$.
4. Compute $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$.
5. Compute $(x_1, y_1) = u_1G + u_2Q$.
6. The signature is valid if $r = x_1 \bmod n$, invalid otherwise.

This is how we know that the verification works the way we want it to:

We have, $s = k^{-1}(e + dr) \bmod n$ which we can rearrange to obtain, $k = s^{-1}(e + dr)$ which is

$$s^{-1}e + s^{-1}rd$$

This is nothing but $we + wrd = (u_1 + u_2d) \pmod n$

We have $u_1G + u_2Q = (u_1 + u_2d)G = kG$ which translates to $v = r$.

Elliptic Curve Pintsov Vanstone Signature (ECPVS)

- Signature scheme using Elliptic Curves
- More efficient than RSA as overhead is extremely low

ECPVS

- Signature Generation

The plaintext message is split into two parts: part C representing the data elements requiring confidentiality and part V representing the data elements presented in plaintext. Both the parts are signed. The signature is generated as follows:

1. A random number k, $1 \le k \le n-1$ is chosen.
2. Calculate the point R on the curve (R = kG).
3. Use point R and a symmetric encryption algorithm to get $e = T_R(C)$.
4. Calculate a variable d such that $d = HASH(e \,||\, I_A \,||\, V)$

where I_A is the identity of the mailer terminal.

5. Now calculate the other part of the signature s as follows: $s = ad + k (\bmod\ n)$.

The signature pair (s,e) is transmitted together with the portion V of the plaintext.

ECPVS

- Signature Verification

1. Retrieve Q_A (Q_A is mailer A's public key)
2. Calculate the variable $d = HASH(e \,||\, I_A \,||\, V)$ using the same HASH algorithm as the one used for generating the signature.
3. Compute $U = sG – dQ_A$.
4. Recover $C = T_u^{-1}(e)$.
5. Run a redundancy test on C. If the test fails, discard the message. Else, the plaintext is recovered.

We have, $s = ad + k$. Multiply by base point G to obtain $sG = adG + kG$ which is $dQ_A + R$

Therefore, $R = sG – dQ_A$ which is U. Comparing the decrypted versions, m and m' obtained using U and R, we ascertain the validity of the signature

Elliptic Curve Diffie-Hellman (ECDH)

- Elliptic curve variant of the key exchange Diffie-Hellman protocol.
- Decide on domain parameters and come up with a Public/Private key pair
- To obtain the private key, the attacker needs to solve the discrete log problem

ECDH

- How the key exchange takes place:

  1. Alice and Bob publicly agree on an elliptic curve E over a large finite field F and a point P on that curve.
  2. Alice and Bob each privately choose large random integers, denoted a and b
  3. Using elliptic curve point-addition, Alice computes aP on E and sends it to Bob. Bob computes bP on E and sends it to Alice.
  4. Both Alice and Bob can now compute the point abP Alice by multiplying the received value of bP by her secret number a and Bob vice-versa.
  5. Alice and Bob agree that the x coordinate of this point will be their shared secret value.

Pros and Cons

- Pros
  - Shorter Key Length
    - Same level of security as RSA achieved at a much shorter key length
  - Better Security
    - Secure because of the ECDLP
    - Higher security per key-bit than RSA
  - Higher Performance
    - Shorter key-length ensures lesser power requirement – suitable in wireless sensor applications and low power devices
    - More computation per bit but overall lesser computational expense or complexity due to lesser number of key bits

Pros and Cons

- Cons
  - Relatively newer field
    - Idea prevails that all the aspects of the topic may not have been explored yet – possibly unknown vulnerabilities
    - Doesn't have widespread usage
  - Not perfect
    - Attacks still exist that can solve ECC (112 bit key length has been publicly broken)
    - Well known attacks are the Pollard's Rho attack (complexity $O(\sqrt{n})$), Pohlig's attack, Baby Step,Giant Step etc