Applied Cryptography
CPEG 472/672
Lecture 12A

Instructor: Nektarios Tsoutsos

D-H key agreement with EC

- Recall, in regular D-H we have secret a
  - Alice sends $A = g^a \bmod p$ to Bob
  - Shared secret is $g^{ab} \bmod p$
- ECDH uses a fixed point G and d_A, d_B
  - d_A is Alice's random secret number
  - Alice sends $P_A = d_A G$ to Bob
  - d_B is Bob's random secret number
  - Bob sends $P_B = d_B G$ to Alice
  - Shared secret: $d_B P_A = d_A P_B = d_A d_B G$
  - Relies on ECDLP hardness assumption

EC Digital Signature Algorithm

- ECDSA used in many applications
  - Bitcoin, TLS, SSH etc.
- Signature Generation & Verification
  - Curve selection, base point G are known
    - The multiplicative order n of G is known
    - If n is the order, nG equals the identity element
  - Generate signatures with private value d
  - Verify signatures with the public point $P = dG$

ECDSA Signatures

- Signature Generation
  - h is the message hash (e.g., use SHA-256)
    - h must be between 1 an $n - 1$
  - Pick k between 1 an $n - 1$
  - Compute point $\mathrm{V} = kG$ with coordinates (x, y)
  - Compute $r = x \bmod n$
  - Compute $s = (h + rd)k^{-1} \bmod n$
  - The signature of h is (r, s)
- With 256-bit coordinates, (r, s) is 512 bits

ECDSA Signatures

- Signature Verification
  - Recall, we defined $s = (h + rd)k^{-1} \bmod n$
  - Compute $u = h \cdot s^{-1} \bmod n$, $v = r \cdot s^{-1} \bmod n$
  - Compute point $Q = uG + vP$
    - Recall, P is signer's public point (i.e., $P = dG$)
  - Accept the signature (r, s) as correct for h if the x coordinate of Q equals r
  - Why?
    - $Q = uG + vP = uG + vdG = (u + vd)G = kG$
    - So $Q == V$, so x coordinate of Q should equal r

ECDSA vs RSA

- RSA used for encryption and signatures
  - $y = x^d \bmod N$ is signature of x (priv. key d)
  - Verification: check if $y^e \bmod N$ equals x
    - Simpler than ECDSA
- ECC works with smaller numbers vs RSA
  - Shorter signatures vs RSA
  - Faster signing speed vs RSA (about 150x)
  - Similar verification speed as RSA for similar security level

Encryption with EC

- Integrated Encryption Scheme (ECIES)
  - Hybrid asymmetric/symmetric encryption
  - Based on ECDH key exchange
  - Can encrypt short plaintexts
- ECIES Encryption of message M
  - We know recipients public point $P = dG$
  - Compute point $Q = vG$ for random v (G given)
  - Compute ECDH secret $S = vP$, apply KDF on S
  - Encrypt M using AEAD using key from KDF
  - Send Q, ctxt C and tag T to recipient
- ECIES Decryption: Recover $S = dQ = dvG$,

Choosing a safe curve

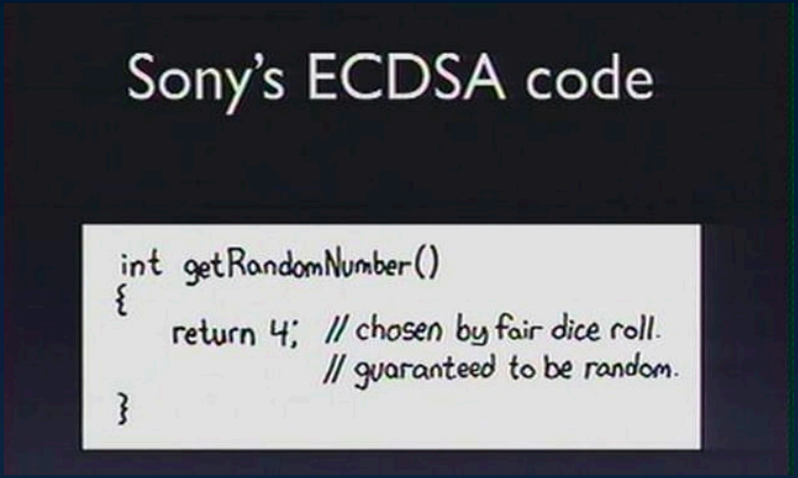
- Choose parameters a,b carefully
  - Some choices are not secure
  - Use standard curves
  - The EC group order must not be product of small numbers (otherwise ECDLP is easy)
  - Prefer curves with unified addition lay
    - Point addition when $P \neq Q$ same as when $P = Q$
    - Using different code affects indistinguishability
    - Best if same formula is used
  - Prefer curves that show how a,b are chosen

Popular curves

- NIST P256 curve
  - Equation: $y^2 = x^3 - 3x + b \bmod p$
  - Prime: $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$
  - Value b: chosen by the NSA
    - A lot of criticism on how b is chosen
    - Explanation: b comes from hashing a constant
- NIST curves P192, P224, P385 and P521
- Curve25519: $y^2 = x^3 + 486662x^2 + x \bmod p$
  - $p = 2^{255} - 19$, clear explanation for 486662
  - Used by Google, Apple etc., fast in software

What can go wrong?

- ECDSA that reuses the same k
  - If two messages use the same k, an attacker can recover signer's secret value d
    - $s_1 = (h_1 + rd)k^{-1} \bmod n$, $s_2 = (h_2 + rd)k^{-1} \bmod n$
    - $k = (h_1 - h_2)/(s_1 - s_2), d = (ks_1 - h_1)/r$
  - Attack on Playstation 3 (2010)



Invalid Curve Attack on ECDH

- Need to validate input point in ECDH
  - Otherwise it is possible to break ECDH
  - Formula for point add doesn't use b
  - Simply adding two points does not confirm that both points come from the same curve
- Invalid Curve Attack
  - Alice sends P_A on a different curve where solving the ECDLP is easy
  - Bob computes the shared secret $d_B P_A$
    - Easy to find the shared secret
- Solution: Check P, Q satisfy curve equation

Hands-on exercises

- ECDH using NIST curves
- ECDSA using P256 NIST curve
- ECIES using P256 NIST curve
- ECDH using Curve25519

Reading for next lecture

- Computing Arbitrary Functions of Encrypted Data: Sections 1 & 2
  - https://crypto.stanford.edu/craig/easy-fhe.pdf