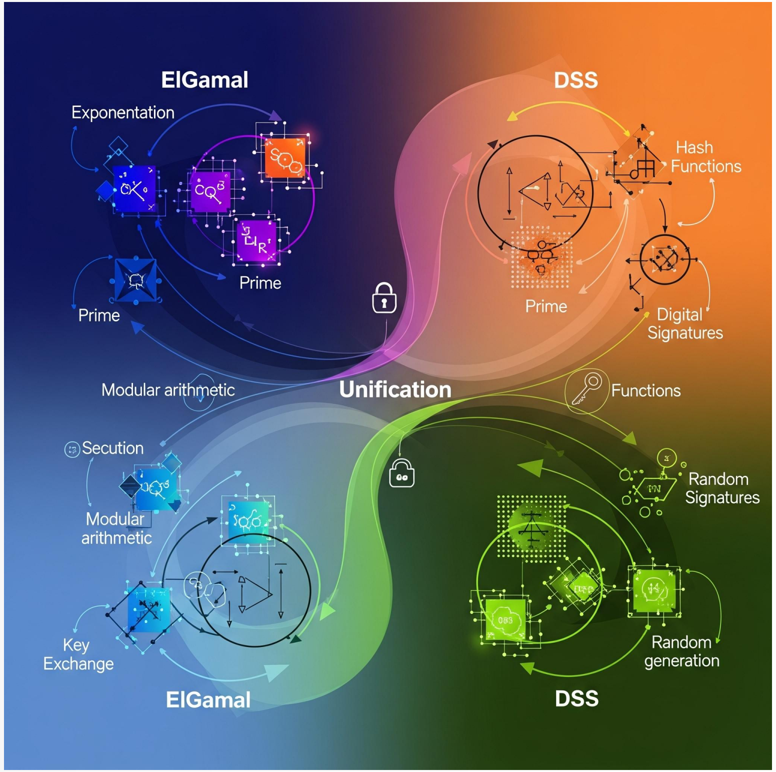
Overview on ElGamal and DSS



Historical note

- Taher Elgamal (b. 1955, Cairo) known for creating the ElGamal encryption and signature schemes. Former Chief Scientist at Netscape, he contributed to SSL/TLS and has held senior roles in major security companies. Often called the “father of SSL”
- Currently working in private industry

ElGamal

- Two different methods was created for encryption (confidentiality) and signature
- Both inspired to Diffie-Hellman

DSS

- The first alternatives to RSA: ElGamal and DSS
- NIST took the ElGamal signature idea (1985), modified it to be more efficient and standardized it as DSA in FIPS 186 (1994)
  - DSS is the NIST standard based on DSA algorithm
- More modern alternatives exist

ElGamal encryption and signature are different

- Each of them is used for one goal only and cannot be used for the other
  - different from RSA

ElGamal encryption (simplified)

- Based on DH and consists of three components: key generator, encryption algorithm, decryption algorithm
- Alice publishes p, $g \in Z_p^*$ as public parameters
    - g = generator of a cyclic group of order p
- Alice chooses x as a private key and publishes g^x mod p as a public key
    - x chosen at random in {0, 1, … , p-1}
- Encryption (Bob, for Alice) of $m \in Z_p^*$ by sending (g^y mod p, mg^{xy} mod p)
    - y chosen per-message at random in {0, 1, … , p-1}

ElGamal decryption

- Alice computes $(g^y)^x \bmod p = g^{xy} \bmod p$, then computes $(g^{xy})^{-1} \bmod p$ for obtaining $mg^{xy}(g^{xy})^{-1} \bmod p = m$

- Requires two exponentiations per each block transmitted

- The involved math is not trivial

ElGamal signature

Key generation

- Pick a prime p of length 1024 bits such that DL in Z_p^* is hard
- Let g be a generator of Z_p^*
- Pick x in [2, p-2] at random
- Compute $y = g^x \bmod p$
- Public key: (p, g, y)
- Private key: x

ElGamal signature

Signing M [a per-message public/private key pair (r, k) is also generated]

- Hash: Let m = H(M)
- Pick k in [1, p-2] relatively prime to p-1 at random
- Compute $r = g^k \bmod p$
- Compute $s = (m-rx)k^{-1} \bmod (p-1)$
    - if s is zero, restart
- Output signature (r, s)

Note: use mod (p – 1) to compute the inverse of k

ElGamal signature

Verify M, r, s, p, k

- Compute $m = H(M)$
- Accept if $(0 < r < p) \land (0 < s < p-1) \land (y^r r^s = g^m) \bmod p$, else reject
- What's going on?
- Since $s = (m\text{-}rx)k^{-1} \bmod (p\text{-}1)$, then $sk + rx = m$. Now $r = g^k$ so $r^s = g^{ks}$, and $y = g^x$ so $y^r = g^{rx}$, implying $y^r r^s = g^{rx} \cdot g^{ks} = g^m$

ElGamal vs RSA

1. **Computation**
    1. ElGamal: Requires multiple modular exponentiations for signing and verification, which can be computationally more intensive
    2. RSA: Generally, more efficient for verification, as it only needs one exponentiation, making it faster in scenarios where quick verification is needed
2. **Signature Size**
    1. ElGamal: Signatures are larger due to two components (r, s), leading to higher storage and transmission costs
    2. RSA: Single component signatures result in smaller signature sizes, reducing data overhead
3. **Security Level**
    1. ElGamal: based on the discrete logarithm problem, providing strong security but susceptible to chosen-message attacks
    2. RSA: Based on integer factorization; both offer comparable security levels, but RSA is more widely adopted
4. **Concluding**
    1. RSA is generally preferred in environments prioritizing fast verification and smaller signatures
    2. ElGamal may offer theoretical security advantages, but at a cost of increased computational load

DSS

- NIST, FIPS PUB 186
- DSS uses a cryptographic hash function H (originally SHA-1) and DSA as signature
- DSA inspired by ElGamal

DSA - preparation

- Let p be an L bit prime such that the discrete log (DL) problem mod p is intractable
- Let q be a 160-bit prime that divides p – 1
  $p = j \cdot q + 1$
  - now $|p|$ = 1024 to 3072 bits and $|q|$ = 160 to 256 bits
- Let α be a q-th root of 1 modulo p
  $\alpha = 1^{1/q} \bmod p$, or $\alpha^q = 1 \bmod p$

?How do we compute α

Computing α

- take a random number h s.t. $1 < h < p - 1$ and compute $g = h^{(p-1)/q} \bmod p = h^j \bmod p$
- if $g = 1$ try a different h
  - things would be insecure
- it holds $g^q = h^{p-1}$
- by Fermat's theorem $h^{p-1} = 1 \bmod p$
  - p is prime and h is by construction not a multiple of p
- choose $\alpha = g$

DSA- execution

- p prime, q prime, $p - 1 = 0 \bmod q$, $\alpha = 1^{(1/q)} \bmod p$
- Private key: random and secret s, $1 \le s \le q-1$
- Public key: $(p, q, \alpha, y = \alpha^s \bmod p)$
- Signature on message M
    1. Choose a random $1 \le k \le q-1$, secret
    2. $P1 = (\alpha^k \bmod p) \bmod q$
    3. $P2 = (H(M) + s \cdot P1)\, k^{-1} \bmod q$
    4. Signature (P1, P2)
- Note that P1 does not depend on M (preprocessing)
- P2 is fast to compute

DSS - verification

- $e1 = H(M)\ (P2)^{-1} \bmod q$
- $e2 = (P1)\ (P2)^{-1} \bmod q$
- ACCEPT signature if $(\alpha^{e1}\, y^{e2} \bmod p) \bmod q = P1$
    - why? see next slide

DSS – correctness

Accept if $(\alpha^{e1} y^{e2} \bmod p) \bmod q = P1$

$e1 = H(M)/P2 \bmod q$

$e2 = P1/P2 \bmod q$

Proof

1. Definition of P2 implies $H(M)/P2 + s\, P1/P2 = k \bmod q$
2. Replace here e1 and e2: $e1 + s \cdot e2 = k \bmod q$
3. Definition of $y = \alpha^s \bmod p$ implies $\alpha^{e1}y^{e2} \bmod p = \alpha^{e1}\alpha^{se2} \bmod p = \alpha^{(H(M)/P2+sP1/P2) \bmod q} \bmod p = \alpha^{k+cq} \bmod p = \alpha^k \bmod p$ (since $\alpha^q = 1$)
4. Execution of mod q implies $(\alpha^{e1}y^{e2} \bmod p) \bmod q = (\alpha^k \bmod p) \bmod q = P1$

DSS – security

- Private key s is not revealed, and DSS cannot be forged without knowing it
- Use of a random number for signing - not revealed (k)
  - There are no duplicates of the same signature (even if same messages)
  - If k is known, then you can compute s mod q = s (s is
    chosen < q)
    - make s explicit from P2 (see next slide)
  - Two messages signed with same k can reveal the value k and therefore s mod q
    - 2 equations (P2' and P2''), 2 unknowns (s and k)
- There exist other sophisticated attacks depending on implementation

If adversary knows k...

$P2 = (H(M) + s \cdot P1)\, k^{-1} \bmod q$
$P2 \cdot k = (H(M) + s \cdot P1) \bmod q$
$(P2 \cdot k - H(M))\, P1^{-1} = s \bmod q = s$ (since $s < q$)

then adv knows s
now adv. wants to sign M′

- $P1 = (\alpha^k \bmod p) \bmod q$ (independent on M′)
- $P2 = ((H(M') + s \cdot P1)k^{-1}) \bmod q$

DSS: efficiency

- Finding two primes p and q such that $p - 1 = 0 \bmod q$ is not easy and takes time
- p and q are public: they can be used by many persons
- DSS slower than RSA in signature verification
- DSS and RSA same speed for signing (DSS faster if you use preprocessing)
- DSS requires random numbers: not always easy to generate

RSA vs. DSS

Feature	RSA	DSS
Purpose	Encryption & signatures	Signatures only
Math basis	Integer factorization	Discrete logarithms
Speed	Signing slower, verification faster	Signing faster, verification slower
Signature size	≈ key size	Smaller
Randomness	None required for signing	Needs fresh random k per signature
Quantum safety	Not safe (Shor's)	Not safe (Shor's)