Chapter 10

Other Public-Key Cryptosystems

Diffie-Hellman Key Exchange

- First published public-key algorithm
- A number of commercial products employ this key exchange technique
- Purpose is to enable two users to securely exchange a key that can then be used for subsequent symmetric encryption of messages
- The algorithm itself is limited to the exchange of secret values
- Its effectiveness depends on the difficulty of computing discrete logarithms

Diffie-Hellman Key Exchange

- a public-key distribution scheme
  - 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 Setup

- all users agree on global parameters:
  - large prime integer or polynomial q
  - a being a primitive root mod q
- each user (eg. A) generates their key
  - chooses a secret key (number): $x_A < q$
  - compute their **public key**: $y_A = a^{x_A} \bmod q$
- each user makes public that key y_A

Diffie-Hellman Key Exchange

- shared session key for users A & B is K_{AB}:

$$K_{AB} = a^{x_A . x_B} \bmod q$$
$$= y_A^{x_B} \bmod q \quad \text{(which } \mathbf{B} \text{ can compute)}$$
$$= y_B^{x_A} \bmod q \quad \text{(which } \mathbf{A} \text{ can compute)}$$

- K_{AB} is used as session key in private-key encryption scheme between Alice and Bob
- if Alice and Bob subsequently communicate, they will have the **same** key as before, unless they choose new public-keys
- attacker needs an x, must solve discrete log

Diffie-Hellman Example

- users Alice & Bob who wish to swap keys:
- agree on prime $q=353$ and $a=3$
- select random secret keys:
  - A chooses $x_A=97$, B chooses $x_B=233$
- compute respective public keys:
  - $y_A=3^{97} \bmod 353 = 40$ (Alice)
  - $y_B=3^{233} \bmod 353 = 248$ (Bob)
- compute shared session key as:
  - $K_{AB}= y_B^{x_A} \bmod 353 = 248^{97} = 160$ (Alice)
  - $K_{AB}= y_A^{x_B} \bmod 353 = 40^{233} = 160$ (Bob)

Key Exchange Protocols

- users could create random private/public D-H keys each time they communicate
- users could create a known private/public D-H key and publish in a directory, then consulted and used to securely communicate with them
- both of these are vulnerable to a meet-in-the-Middle Attack
- authentication of the keys is needed

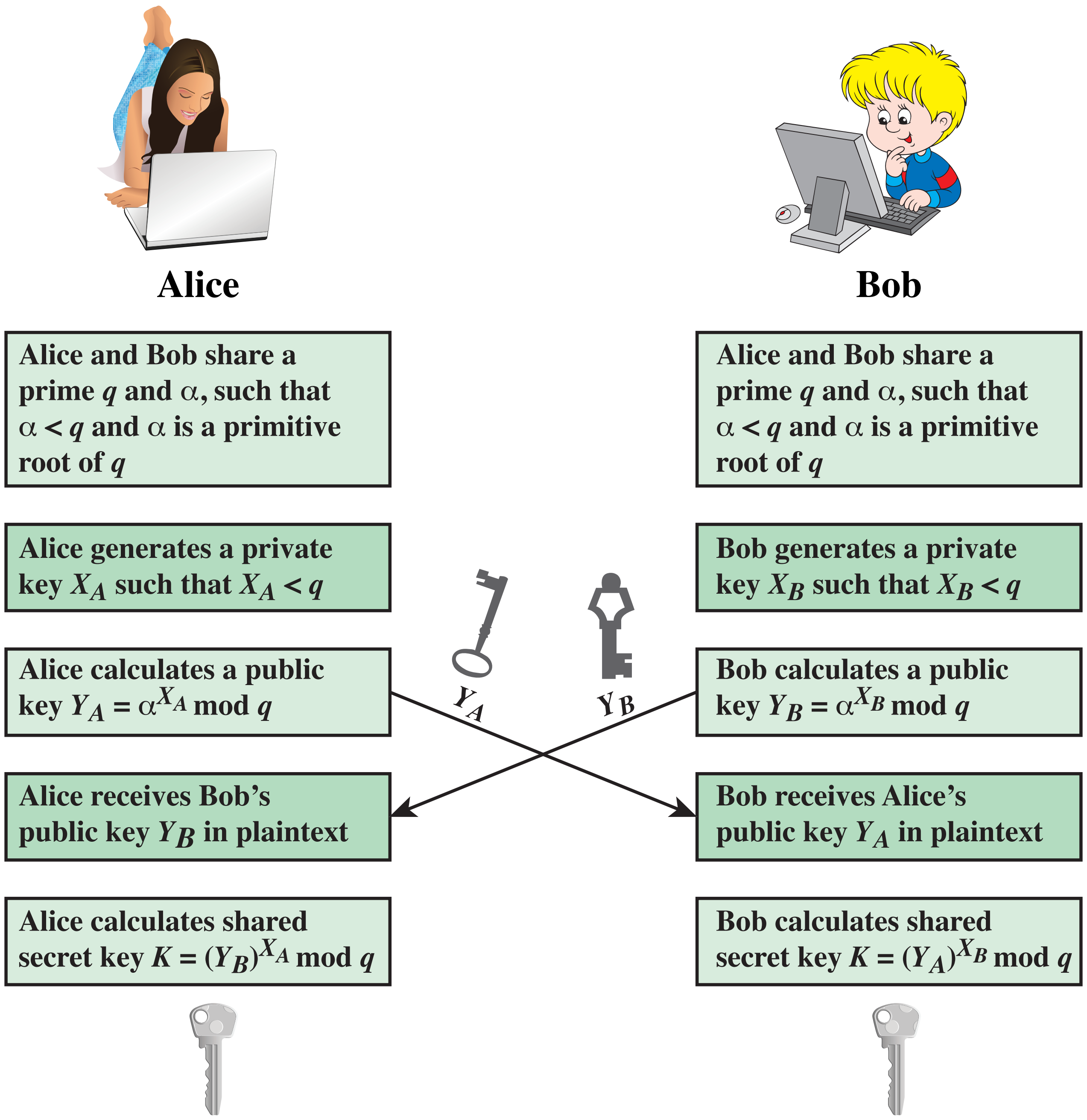


Figure 10.1 Diffie-Hellman Key Exchange

Man-in-the-Middle Attack

1. Darth prepares by creating two private / public keys
2. Alice transmits her public key to Bob
3. Darth intercepts this and transmits his first public key to Bob. Darth also calculates a shared key with Alice
4. Bob receives the public key and calculates the shared key (with Darth instead of Alice)
5. Bob transmits his public key to Alice
6. Darth intercepts this and transmits his second public key to Alice. Darth calculates a shared key with Bob
7. Alice receives the key and calculates the shared key (with Darth instead of Bob)

- Darth can then intercept, decrypt, re-encrypt, forward all messages between Alice & Bob

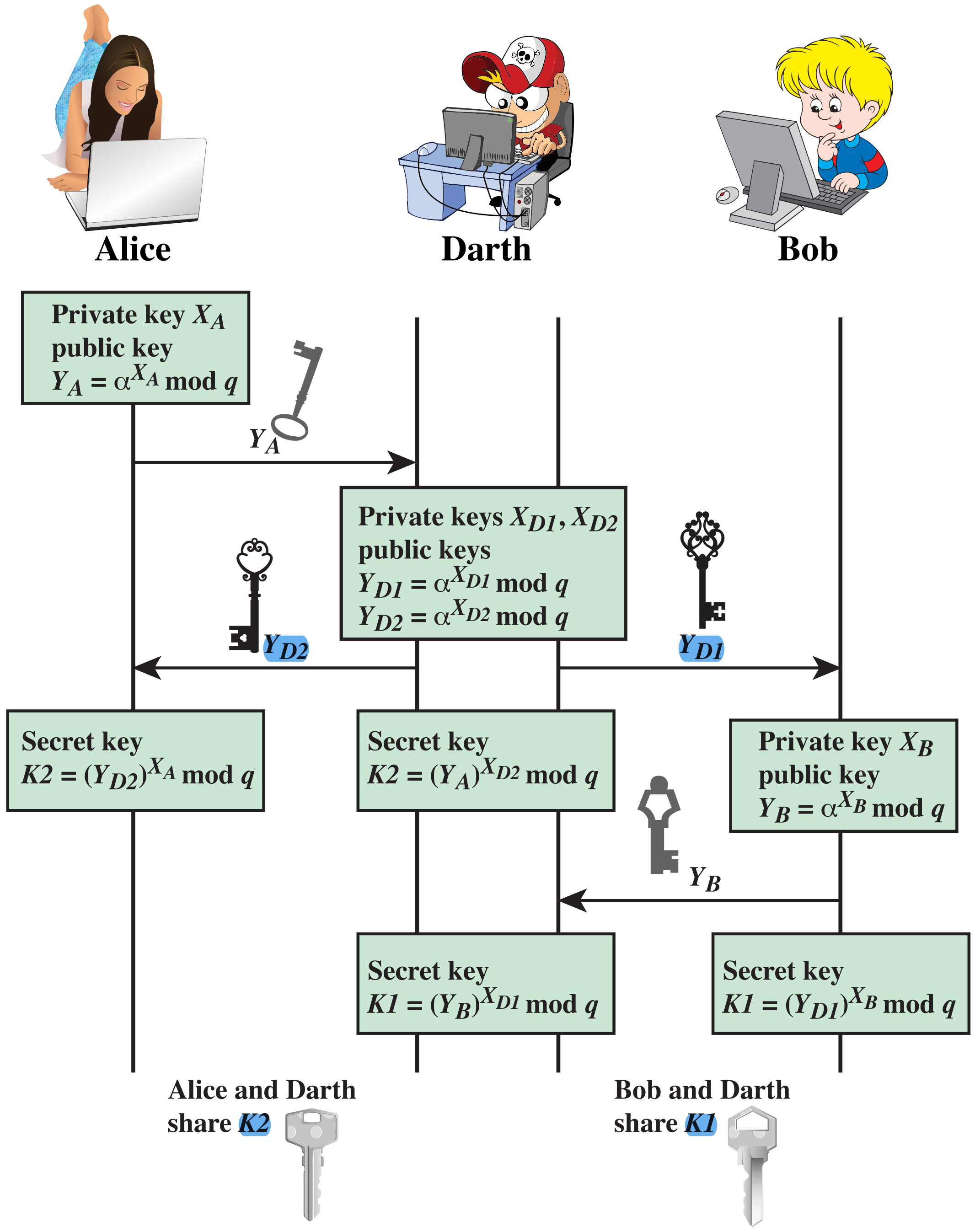


Figure 10.2 Man-in-the-Middle Attack

ElGamal Cryptography

Announced in 1984 by T. Elgamal

Public-key scheme based on discrete logarithms closely related to the Diffie-Hellman technique

Used in the digital signature standard (DSS) and the S/MIME e-mail standard

Global elements are a prime number q and a which is a primitive root of q

Security is based on the difficulty of computing discrete logarithms

Global Public Elements

q	prime number
α	$\alpha < q$ and α a primitive root of q

Key Generation by Alice

Select private X_A	$X_A < q - 1$
Calculate Y_A	$Y_A = \alpha^{X_A} \bmod q$
Public key	$\{q, \alpha, Y_A\}$
Private key	X_A

Encryption by Bob with Alice's Public Key

Plaintext:	$M < q$
Select random integer k	$k < q$
Calculate K	$K = (Y_A)^k \bmod q$
Calculate C_1	$C_1 = \alpha^k \bmod q$
Calculate C_2	$C_2 = KM \bmod q$
Ciphertext:	(C_1, C_2)

Decryption by Alice with Alice's Private Key

Ciphertext:	(C_1, C_2)
Calculate K	$K = (C_1)^{X_A} \bmod q$
Plaintext:	$M = (C_2K^{-1}) \bmod q$

Figure 10.3 The ElGamal Cryptosystem

ElGamal Cryptography

- public-key cryptosystem related to D-H
- so uses exponentiation in a finite (Galois)
- with security based difficulty of computing discrete logarithms, as in D-H
- each user (eg. A) generates their key
  - chooses a secret key (number): $1 < x_A < q-1$
  - compute their **public key**: $y_A = a^{x_A} \bmod q$

ElGamal Message Exchange

- Bob encrypt a message to send to A computing
  - represent message M in range $0 <= M <= q-1$
    - longer messages must be sent as blocks
  - chose random integer k with $1 <= k <= q-1$
  - compute one-time key $K = y_A^{k} \bmod q$
  - encrypt M as a pair of integers (C_1, C_2) where
    - $C_1 = a^{k} \bmod q$; $C_2 = KM \bmod q$
- A then recovers message by
  - recovering key K as $K = C_1^{x_A} \bmod q$
  - computing M as $M = C_2 K^{-1} \bmod q$
- a unique k must be used each time
  - otherwise result is insecure

ElGamal Example

- use field GF(19) $q=19$ and $a=10$
- Alice computes her key:
  - A chooses $x_A=5$ & computes $y_A=10^5 \bmod 19 = 3$
- Bob send message $m=17$ as $(11,5)$ by
  - chosing random $k=6$
  - computing $K = y_A^k \bmod q = 3^6 \bmod 19 = 7$
  - computing $C_1 = a^k \bmod q = 10^6 \bmod 19 = 11;$
    $C_2 = KM \bmod q = 7.17 \bmod 19 = 5$
- Alice recovers original message by computing:
  - recover $K = C_1^{x_A} \bmod q = 11^5 \bmod 19 = 7$
  - compute inverse $K^{-1} = 7^{-1} = 11$
  - recover $M = C_2\ K^{-1} \bmod q = 5.11 \bmod 19 = 17$

Elliptic Curve Arithmetic

- Most of the products and standards that use public-key cryptography for encryption and digital signatures use RSA
  - The key length for secure RSA use has increased over recent years and this has put a heavier processing load on applications using RSA
- Elliptic curve cryptography (ECC) is showing up in standardization efforts including the IEEE P1363 Standard for Public-Key Cryptography
- Principal attraction of ECC is that it appears to offer equal security for a far smaller key size

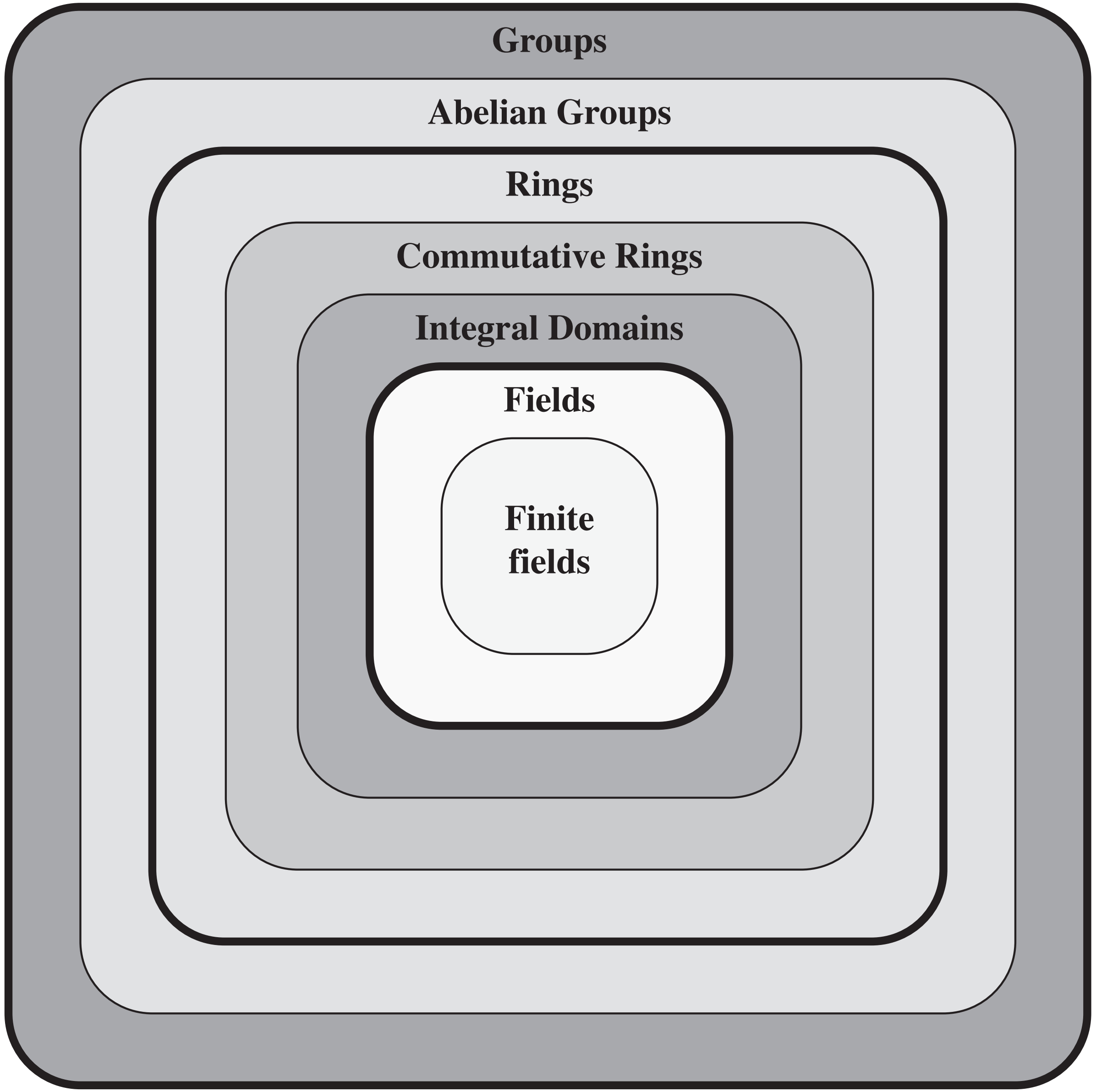


Figure 5.1 Groups, Rings, and Fields

Abelian Group

- A set of elements with a binary operation, denoted by $\bullet$, that associates to each ordered pair (a, b) of elements in G an element $(a \bullet b)$ in G, such that the following axioms are obeyed:

(A1) Closure: If a and b belong to G, then $a \bullet b$ is also in G

(A2) Associative: $a \bullet (b \bullet c) = (a \bullet b) \bullet c$ for all a, b, c in G

(A3) Identity element: There is an element e in G such that $a \bullet e = e \bullet a = a$ for all a in G

(A4) Inverse element: For each a in G there is an element a' in G such that $a \bullet a' = a' \bullet a = e$

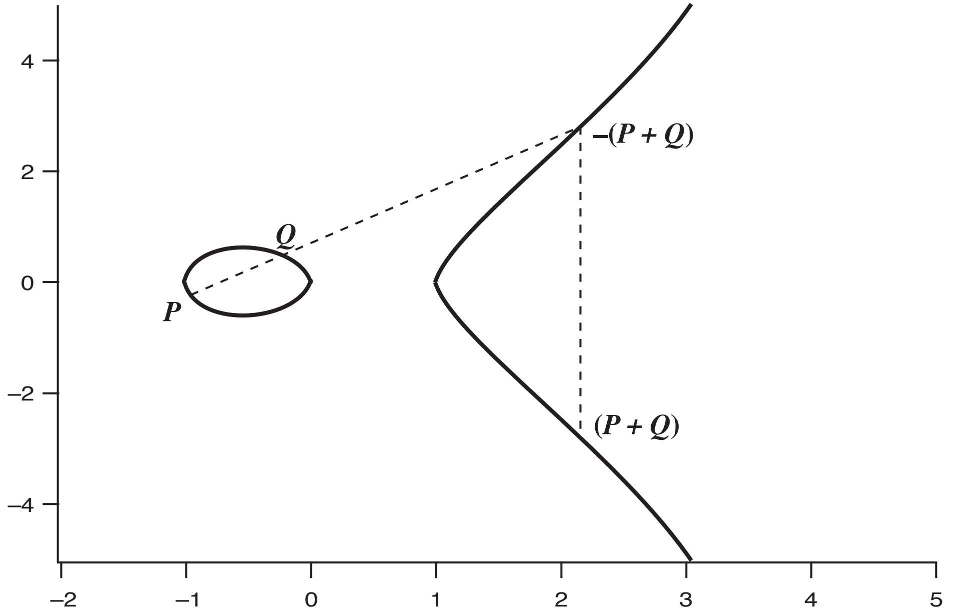
(A5) Commutative: $a \bullet b = b \bullet a$ for all a, b in G

Real Elliptic Curves

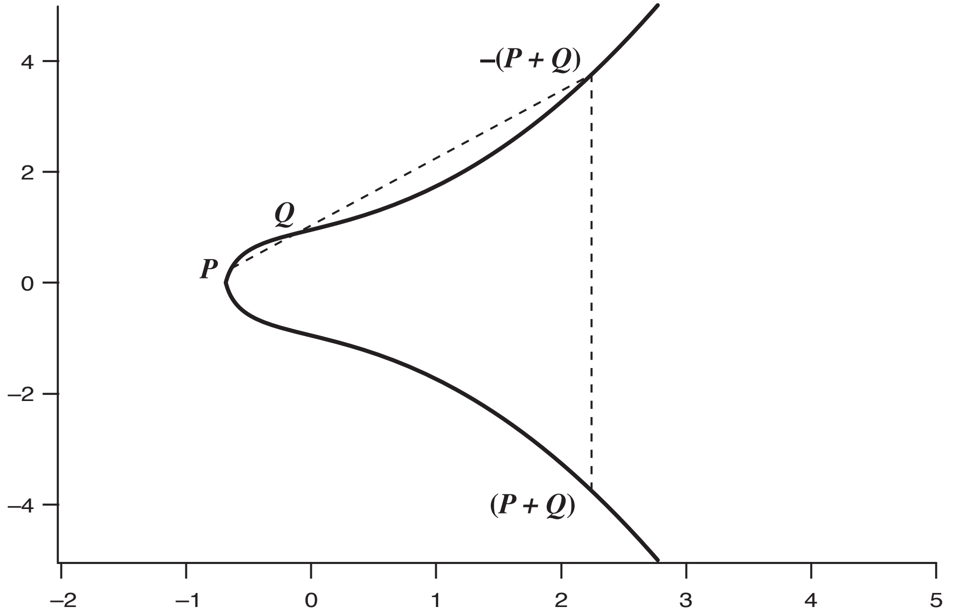
- an elliptic curve is defined by an equation in two variables x & y, with coefficients
- consider a cubic elliptic curve of form
  - $y^2 = x^3 + ax + b$
  - where x,y,a,b are all real numbers
  - also define zero point O
- consider set of points E(a,b) that satisfy
- have addition operation for elliptic curve
  - geometrically sum of P+Q is reflection of the intersection R

Geometric Description of Addition

- Additive identity O (zero point): $P + O = P$
- Negative of point P : $P + (-P) = P - P = O$
  the point with the same x coordinate but negative of the y coordinate.
- Addition of two points P and Q with different x coordinates: draw a straight line between them and find the third point of intersection R. We define to be the mirror image, with respect to the x axis, of the third point of intersection R. $P + Q = -R$
- To double a point P, draw the tangent line and find the other point of intersection S.
  $P + P = 2P = -S$



(a) $y^2 = x^3 - x$



(b) $y^2 = x^3 + x + 1$

Figure 10.4 Example of Elliptic Curves

Field
Integral domain
Commutative ring
Ring
Abelian group
Group

(A1) Closure under addition:	If a and b belong to S, then $a + b$ is also in S
(A2) Associativity of addition:	$a + (b + c) = (a + b) + c$ for all a, b, c in S
(A3) Additive identity:	There is an element 0 in R such that $a + 0 = 0 + a = a$ for all a in S
(A4) Additive inverse:	For each a in S there is an element $-a$ in S such that $a + (-a) = (-a) + a = 0$
(A5) Commutativity of addition:	$a + b = b + a$ for all a, b in S
(M1) Closure under multiplication:	If a and b belong to S, then ab is also in S
(M2) Associativity of multiplication:	$a(bc) = (ab)c$ for all a, b, c in S
(M3) Distributive laws:	$a(b + c) = ab + ac$ for all a, b, c in S $(a + b)c = ac + bc$ for all a, b, c in S
(M4) Commutativity of multiplication:	$ab = ba$ for all a, b in S
(M5) Multiplicative identity:	There is an element 1 in S such that $a1 = 1a = a$ for all a in S
(M6) No zero divisors:	If a, b in S and $ab = 0$, then either $a = 0$ or $b = 0$
(M7) Multiplicative inverse:	If a belongs to S and $a \neq 0$, there is an element a^{-1} in S such that $aa^{-1} = a^{-1}a = 1$

Figure 5.2 Properties of Groups, Rings, and Fields

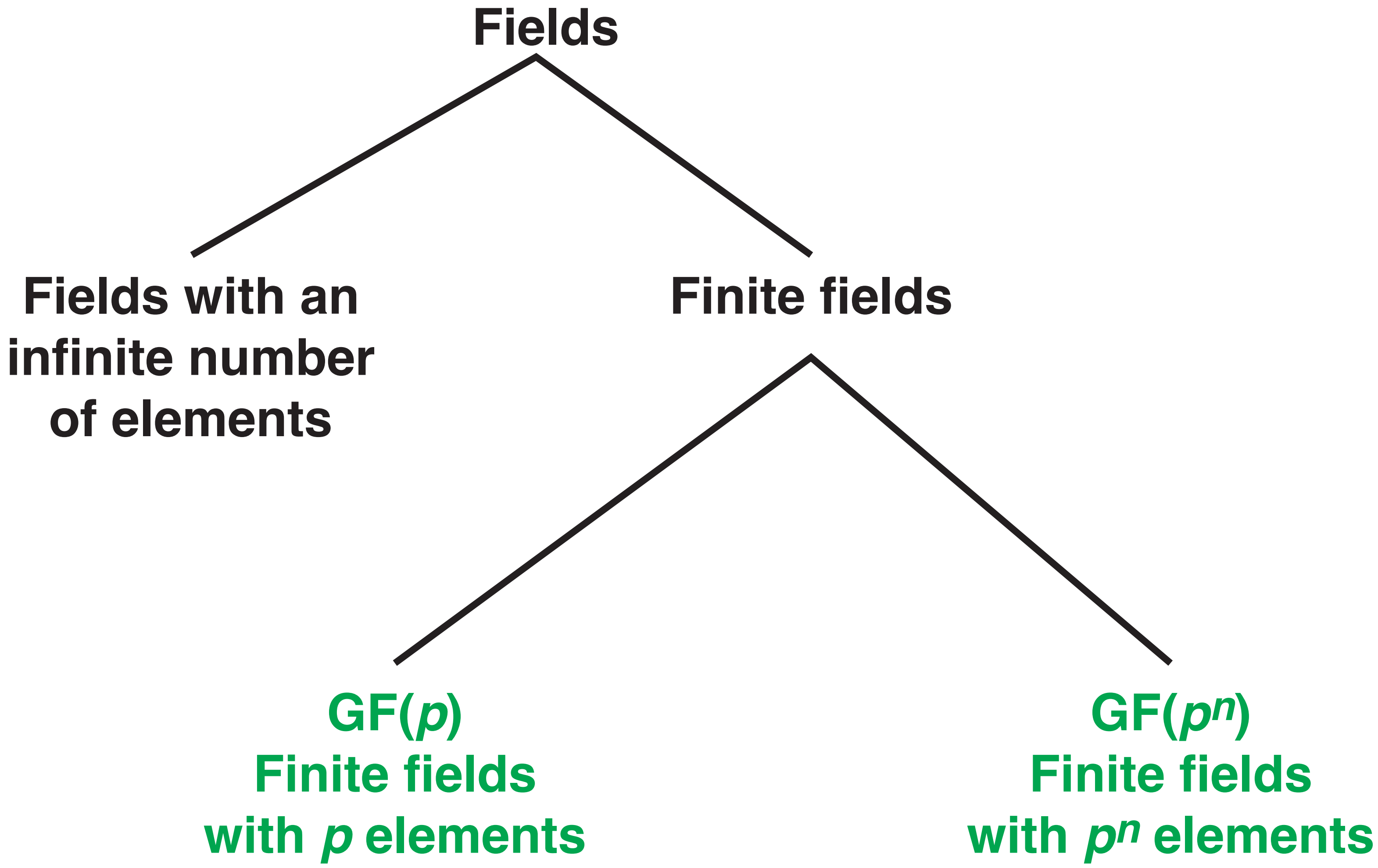
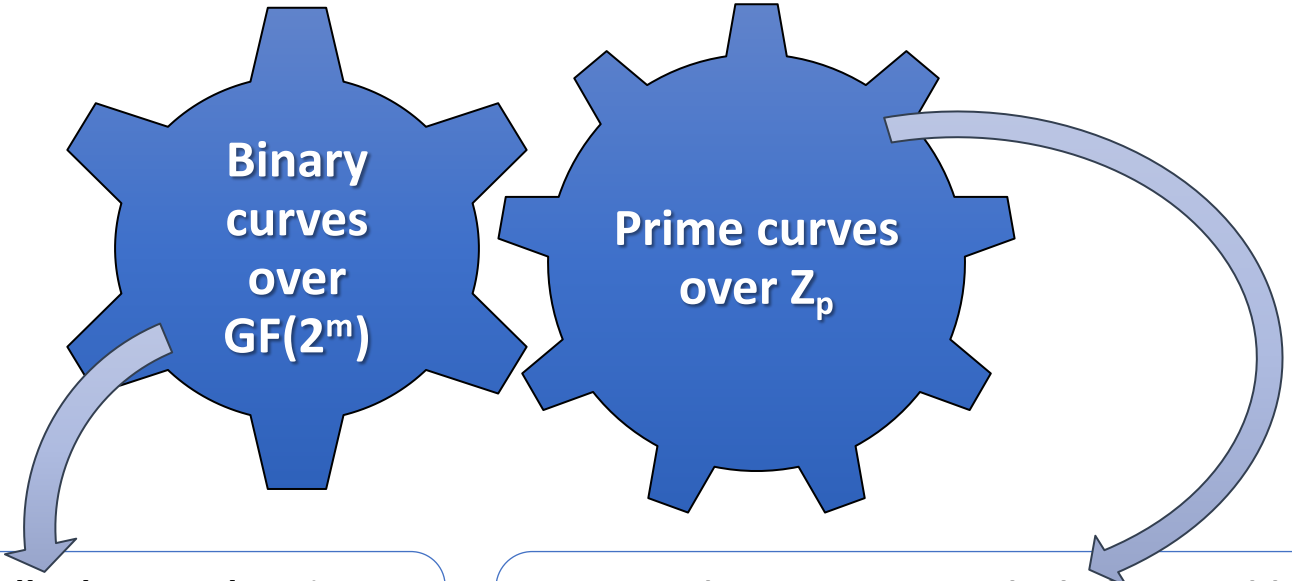


Figure 5.3 Types of Fields

Elliptic Curves Over Z_p

- Elliptic curve cryptography uses curves whose variables and coefficients are finite
- Two families of elliptic curves are used in cryptographic applications:



- Variables and coefficients all take on values in $GF(2^m)$ and in calculations are performed over $GF(2^m)$
- Best for hardware applications

- Use a cubic equation in which the variables and coefficients all take on values in the set of integers from 0 through p-1 and in which calculations are performed modulo p
- Best for software applications

Table 10.1

Points (other than O) on the Elliptic Curve $E_{23}(1, 1)$

(0, 1)	(6, 4)	(12, 19)
(0, 22)	(6, 19)	(13, 7)
(1, 7)	(7, 11)	(13, 16)
(1, 16)	(7, 12)	(17, 3)
(3, 10)	(9, 7)	(17, 20)
(3, 13)	(9, 16)	(18, 3)
(4, 0)	(11, 3)	(18, 20)
(5, 4)	(11, 20)	(19, 5)
(5, 19)	(12, 4)	(19, 18)

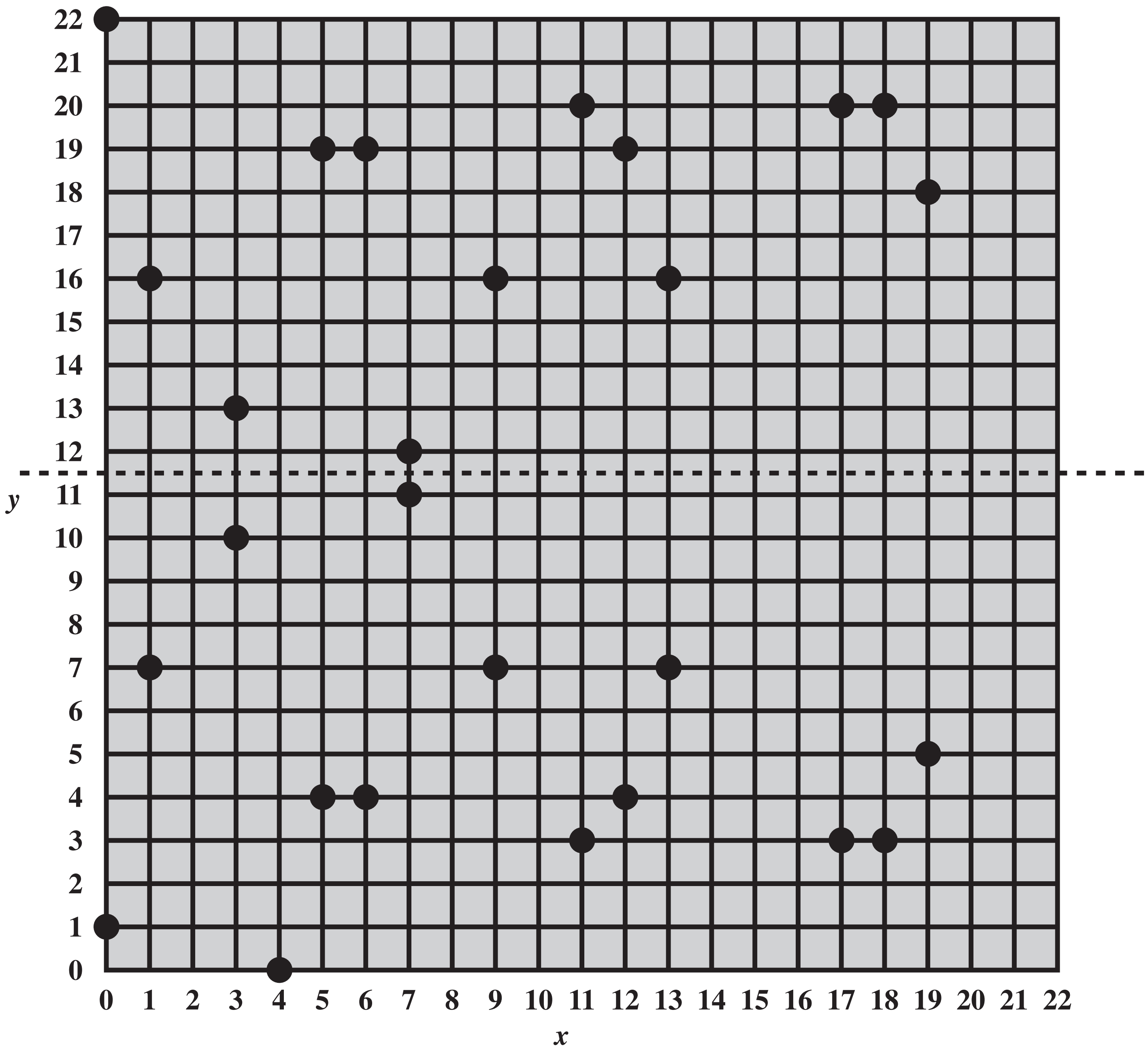


Figure 10.5 The Elliptic Curve $E_{23}(1,1)$

Elliptic Curves Over GF(2^m)

- Use a cubic equation in which the variables and coefficients all take on values in GF(2^m) for some number m
- Calculations are performed using the rules of arithmetic in GF(2^m)
- The form of cubic equation appropriate for cryptographic applications for elliptic curves is somewhat different for GF(2^m) than for Z_p
  - It is understood that the variables x and y and the coefficients a and b are elements of GF(2^m) and that calculations are performed in GF(2^m)

Table 10.2
Points (other than O) on the Elliptic Curve $E_{2^4}(g^4, 1)$

$(0, 1)$	(g^5, g^3)	(g^9, g^{13})
$(1, g^6)$	(g^5, g^{11})	(g^{10}, g)
$(1, g^{13})$	(g^6, g^8)	(g^{10}, g^8)
(g^3, g^8)	(g^6, g^{14})	$(g^{12}, 0)$
(g^3, g^{13})	(g^9, g^{10})	(g^{12}, g^{12})

Elliptic Curve Cryptography (ECC)

- Addition operation in ECC is the counterpart of modular multiplication in RSA
- Multiple addition is the counterpart of modular exponentiation

To form a cryptographic system using elliptic curves, we need to find a "hard problem" corresponding to factoring the product of two primes or taking the discrete logarithm

- $Q=kP$, where Q, P belong to a prime curve
- Is "easy" to compute Q given k and P
- But "hard" to find k given Q, and P
- Known as the elliptic curve logarithm problem

Global Public Elements

$E_q(a, b)$	elliptic curve with parameters a, b, and q, where q is a prime or an integer of the form 2^m
G	point on elliptic curve whose order is large value n

User A Key Generation

Select private n_A	$n_A < n$
Calculate public P_A	$P_A = n_A \times G$

User B Key Generation

Select private n_B	$n_B < n$
Calculate public P_B	$P_B = n_B \times G$

Calculation of Secret Key by User A

$K = n_A \times P_B$

Calculation of Secret Key by User B

$K = n_B \times P_A$

Figure 10.7 ECC Diffie-Hellman Key Exchange

ECC Diffie-Hellman

- can do key exchange analogous to D-H
- users select a suitable curve $E_q(a,b)$
- select base point $G=(x_1,y_1)$
  - with large order n s.t. $nG=O$
- A & B select private keys $n_A<n$, $n_B<n$
- compute public keys: $P_A=n_AG$, $P_B=n_BG$
- compute shared key: $K=n_AP_B$, $K=n_BP_A$
  - same since $K=n_An_BG$
- attacker would need to find K, hard

ECC Encryption/Decryption

- Several approaches using elliptic curves have been analyzed
- Must first encode any message m as a point on the elliptic curve P_m
- Select suitable curve and point G as in Diffie-Hellman
- Each user chooses a private key n_A and generates a public key $P_A = n_A * G$
- To encrypt and send message P_m to B, A chooses a random positive integer k and produces the ciphertext C_m *consisting of the pair of points:*

$$C_m = \{kG, P_m + kP_B\}$$

- To decrypt the ciphertext, B multiplies the first point in the pair by B's secret key and subtracts the result from the second point:

$$P_m + kP_B - n_B(kG) = P_m + k(n_BG) - n_B(kG) = P_m$$

Security of Elliptic Curve Cryptography

- Depends on the difficulty of the elliptic curve logarithm problem
- Fastest known technique is “Pollard rho method”
- Compared to factoring, can use much smaller key sizes than with RSA
- For equivalent key lengths computations are roughly equivalent
- Hence, for similar security ECC offers significant computational advantages

Table 10.3

Comparable Key Sizes in Terms of Computational Effort for Cryptanalysis (NIST SP-800-57)

Symmetric key algorithms	Diffie-Hellman, Digital Signature Algorithm	RSA (size of *n* in bits)	ECC (modulus size in bits)
80	$L = 1024$ $N = 160$	1024	160–223
112	$L = 2048$ $N = 224$	2048	224–255
128	$L = 3072$ $N = 256$	3072	256–383
192	$L = 7680$ $N = 384$	7680	384–511
256	$L = 15{,}360$ $N = 512$	15,360	512+

Note: L = size of public key, N = size of private key

Pseudorandom Number Generation (PRNG) Based on Asymmetric Cipher

- An asymmetric encryption algorithm produces apparently ransom output and can be used to build a PRNG
- Much slower than symmetric algorithms so they're not used to generate open-ended PRNG bit streams
- Useful for creating a pseudorandom function (PRF) for generating a short pseudorandom bit sequence

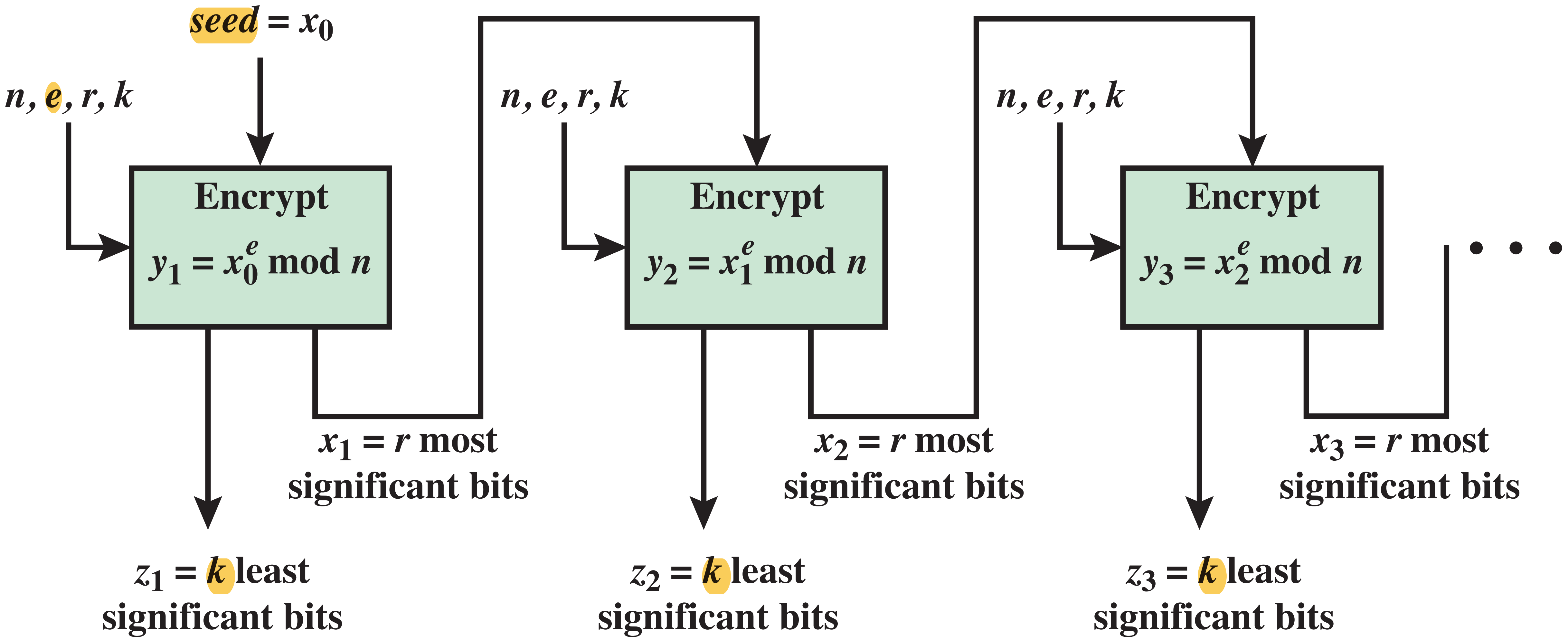


Figure 10.8 Micali-Schnorr Pseudorandom Bit Generator

(ANSI X9.82 and ISO 18031)

Setup Select p, q primes; $n = pq$; $\phi(n) = (p - 1)(q - 1)$. Select e such that $\gcd(e, \phi(n)) = 1$. These are the standard RSA setup selections (see Figure 9.5). In addition, let $N = \lfloor \log_2 n \rfloor + 1$ (the bitlength of n). Select r, k such that $r + k = N$.

Seed Select a random seed x_0 of bitlength r.

Generate Generate a pseudorandom sequence of length $k \times m$ using the loop
for i **from** 1 to m **do**

$$\begin{aligned} y_i &= x_{i-1}^e \bmod n \\ x_i &= r \text{ most significant bits of } y_i \\ z_i &= k \text{ least significant bits of } y_i \end{aligned}$$

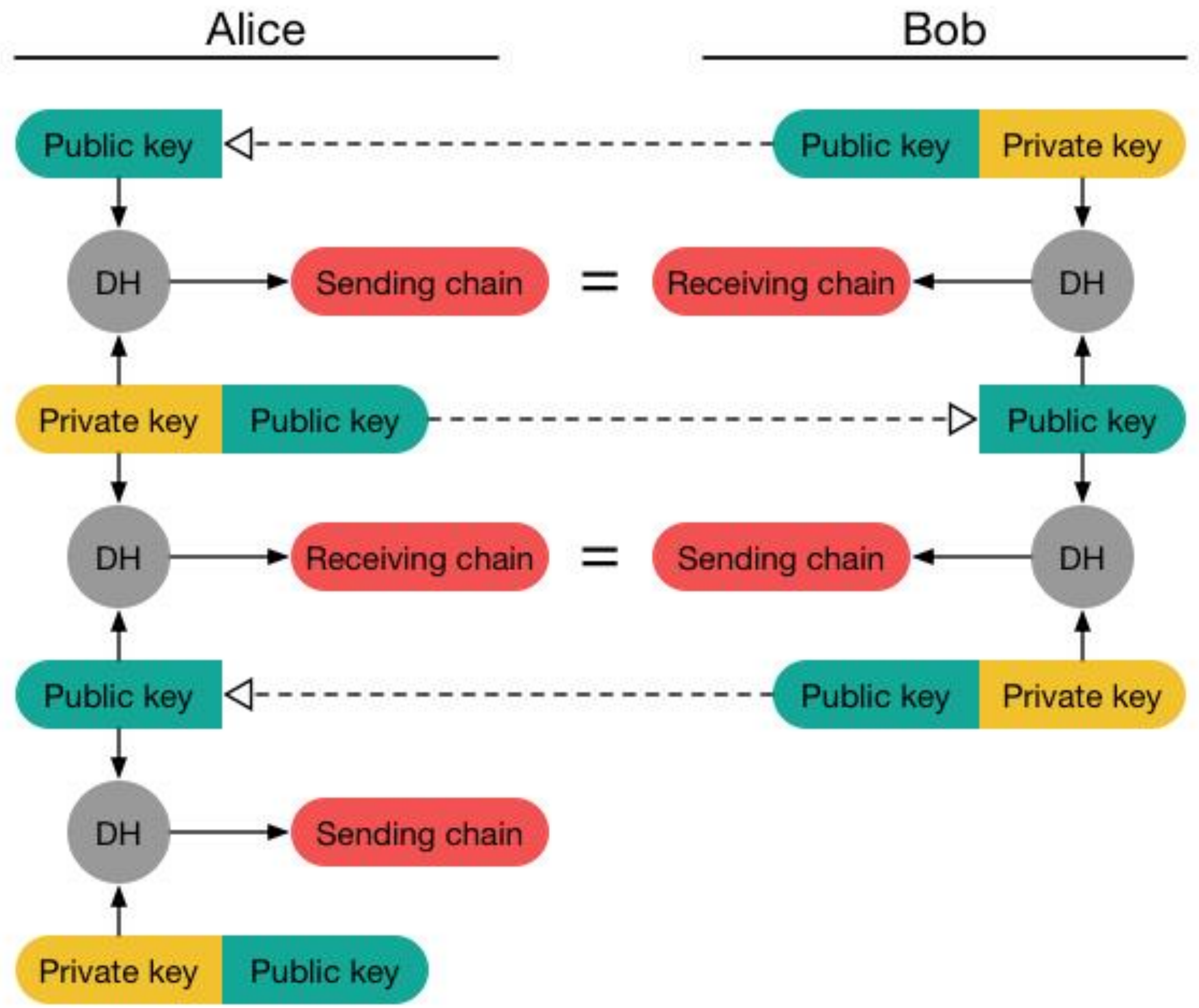
Output The output sequence is $z_1 \| z_2 \| \ldots \| z_m$.

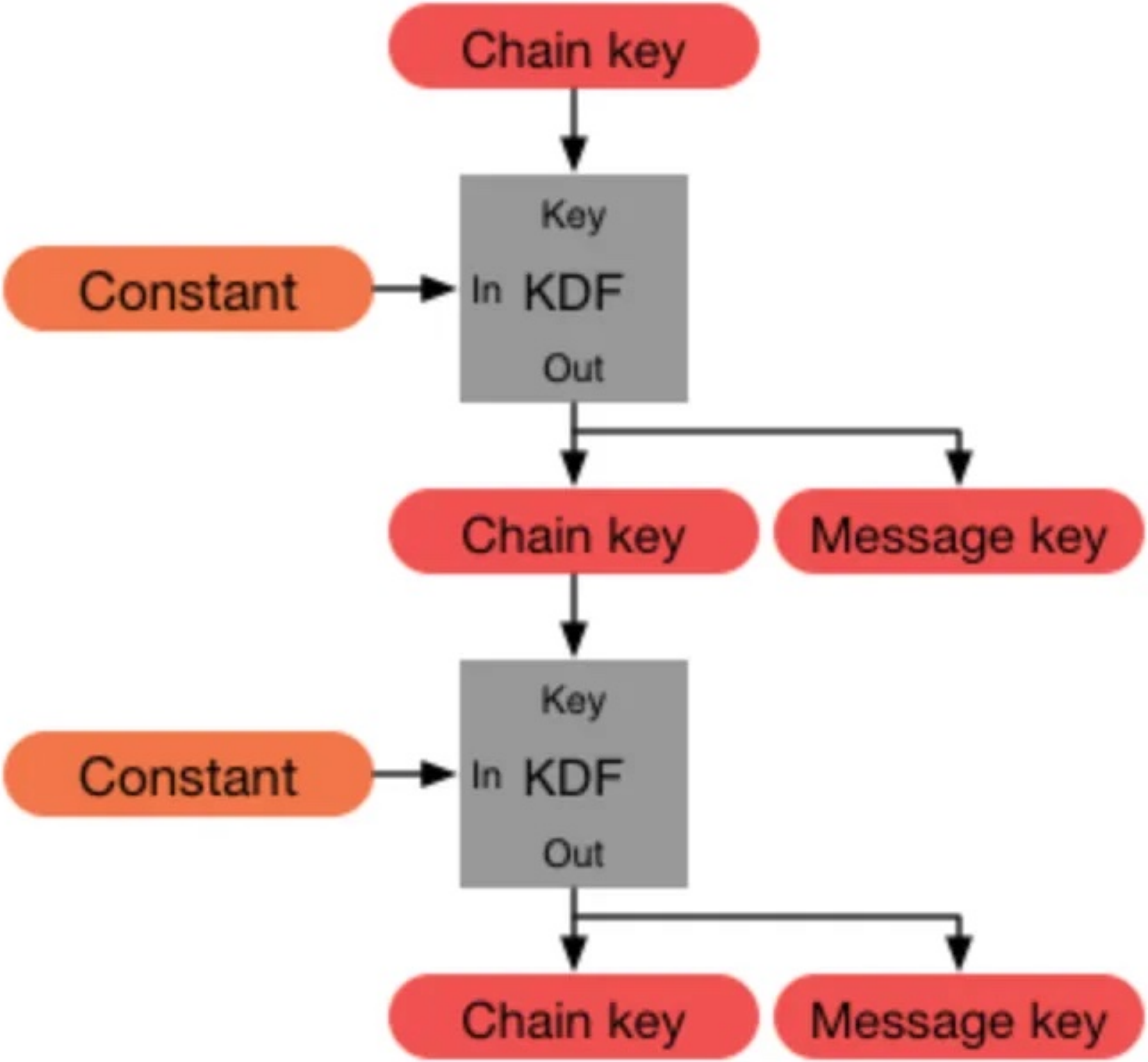
The parameters n, r, e, and k are selected to satisfy the following six requirements.

1. $n = pq$	n is chosen as the product of two primes to have the cryptographic strength required of RSA.
2. $1 < e < \phi(n)$; $\gcd(e, \phi(n)) = 1$	Ensures that the mapping $s \rightarrow s^e \bmod n$ is 1 to 1.
3. $re \geq 2N$	Ensures that the exponentiation requires a full modular reduction.
4. $r \geq 2$ *strength*	Protects against a cryptographic attacks.
5. k, r are multiples of 8	An implementation convenience.
6. $k \geq 8$; $r + k = N$	All bits are used.

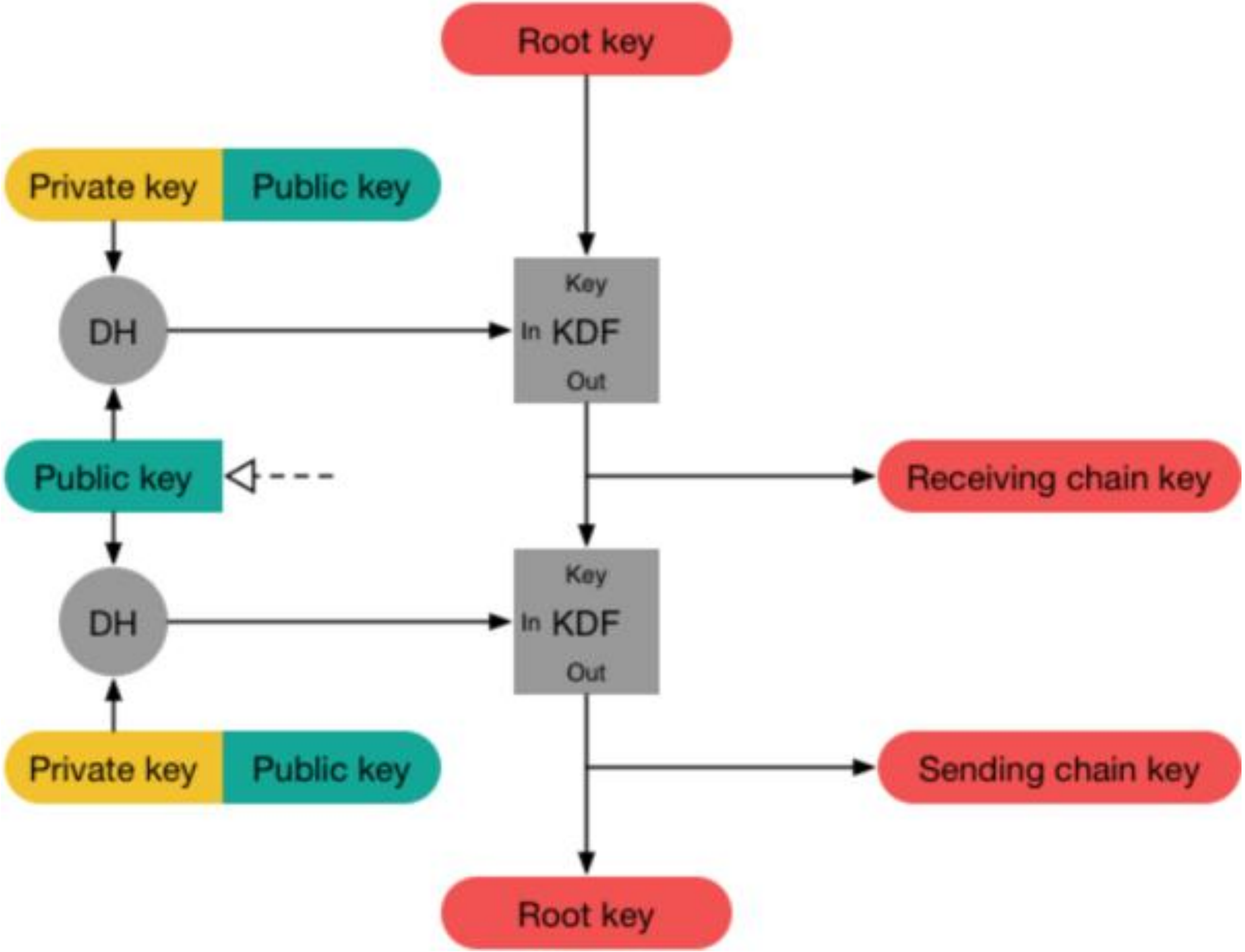
PRNG Based on Elliptic Curve Cryptography

- Developed by the U.S. National Security Agency (NSA)
- Known as dual elliptic curve PRNG (DEC PRNG)
- Recommended in NIST SP 800-90, the ANSI standard X9.82, and the ISO standard 18031
- Has been some controversy regarding both the security and efficiency of this algorithm compared to other alternatives
  - The only motivation for its use would be that it is used in a system that already implements ECC but does not implement any other symmetric, asymmetric, or hash cryptographic algorithm that could be used to build a PRNG


Alice
Bob
Public key
Public key
Private key
DH
Sending chain
=
Receiving chain
DH
Private key
Public key
Public key
DH
Receiving chain
=
Sending chain
DH
Public key
Public key
Private key
DH
Sending chain
Private key
Public key



Chain key
Key
Constant
In
KDF
Out
Chain key
Message key
Key
Constant
In
KDF
Out
Chain key
Message key


Root key
Private key
Public key
DH
Key
In KDF
Out
Public key
Receiving chain key
DH
Key
In KDF
Out
Private key
Public key
Sending chain key
Root key

Summary

- Diffie-Hellman Key Exchange
  - The algorithm
  - Key exchange protocols
  - Man-in-the-middle attack
- Elgamal cryptographic system
- Elliptic curve cryptography
  - Analog of Diffie-Hellman key exchange
  - Elliptic curve encryption/decryption
  - Security of elliptic curve cryptography

- Elliptic curve arithmetic
  - Abelian groups
  - Elliptic curves over real numbers
  - Elliptic curves over Z_p
  - Elliptic curves over $GF(2^m)$
- Pseudorandom number generation based on an asymmetric cipher
  - PRNG based on RSA
  - PRNG based on elliptic curve cryptography