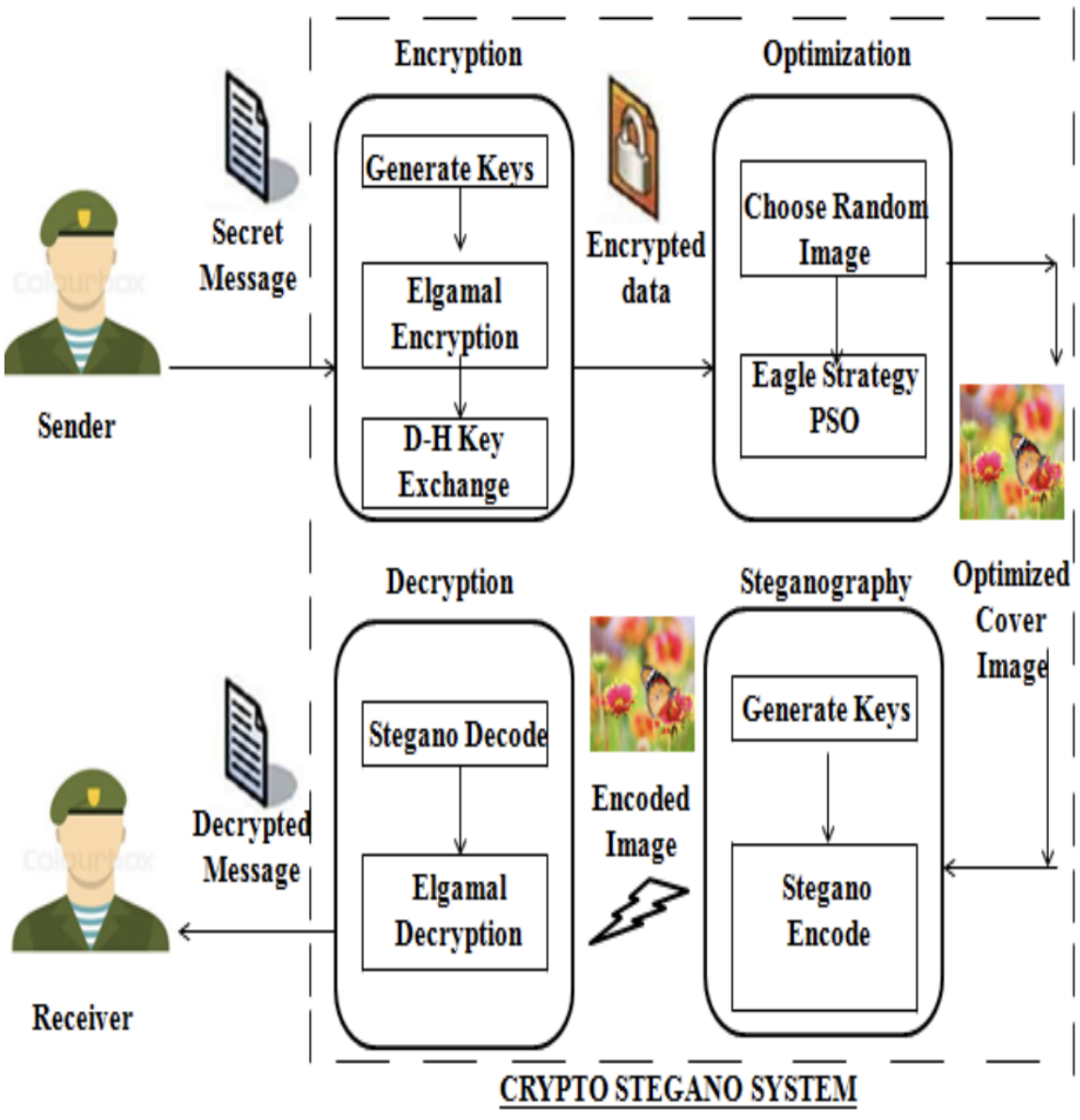
CS4236 Cryptography Theory and Practice Topic 11 - Key agreement, Elgamal, ECC, Bilinear maps...

Hugh Anderson

National University of Singapore
School of Computing

October, 2022

Diffie-Hellman, Elgamal, ECC, bilinear maps...



Outline

1. **More on asymmetric systems**
   - Scheme #1: Key agreement/exchange (KA vs KEM)
   - Elgamal cryptosystem(s)
   - Elliptic Curve Cryptography (ECC)
   - Bilinear maps, 3-party KA and IBE (scheme #2 today)

The story so far ... where are we?

The last 10 weeks: weekly steps we have taken

1. We had a historical/contextual introduction in Session 1.
2. We progressed from perfect *secrecy* to perfect *indistinguishability*.
3. *Perfectly* indistinguishable was relaxed to give *computationally* indistinguishable. An EAV-Secure system was constructed with a PRG.
4. The notion of CPA-secure was developed, and achieved with a PRF.
5. Modes, the “padding oracle”, and CCA-Security were outlined.
6. We looked at cryptographic MACs and *authenticated* encryption.
7. We began looking at hashes, with definitions, games, and applications.
8. We then looked at birthday attacks on hashes, and rainbow tables.
9. We looked at SP networks, and cryptanalysis of them.
10. Last week, we continued with further exploration of the math background for PK systems. We then gave definitions of such systems, and the RSA construction, along with various attacks on it.

Recap: DL vs RSA

The difficult problem in RSA: find e^{th} root in $\mathbb{Z}_N^*$

Given N, e, y, find x s.t. $x^e \bmod N = y$.

Root finding can occur in any group, and there are efficient algorithms when N is prime. The hardness of RSA requires N to be a composite and thus relates to factorization, which only makes sense in a ring with two operators $\times, +$. It is not easy to generalize RSA encryption to other algebraic groups or rings.

The difficult problem in other systems: solve discrete log

Given N, x, y, find e s.t. $x^e \bmod N = y$.

Such problems also naturally occur in any group. It turns out that there many groups whereby these problems seem to be very difficult, and at the same time, with certain useful properties (e.g. bilinear maps). This flexibility makes it attractive to design crypto systems based on DL, DDH or CDH. (there are many variants, e.g. knowledge of exponent, generalized DDH, etc, and many choices of groups, e.g Elliptic Curve, bilinear map, etc).

Reminder: generators in cyclic groups

a	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}
2	2	4	8	5	10	9	7	3	6	1
3	3	9	5	4	1	3	9	5	4	1
4	4	5	9	3	1	4	5	9	3	1
5	5	3	4	9	1	5	3	4	9	1
6	6	3	7	9	10	5	8	4	2	1
7	7	5	2	3	10	4	6	9	8	1
8	8	9	6	4	10	3	2	5	7	1
9	9	4	3	5	1	9	4	3	5	1
10	10	1	10	1	10	1	10	1	10	1

Cyclic groups and subgroups

All elements in $\mathbb{Z}_{11}^*$ can be obtained by successive multiplication by 6. So this is a cyclic group and can be written as $\langle 6 \rangle$. 6 is called a generator.

In general, when p is a prime, for any g in $\mathbb{Z}_p^*$, $\langle g \rangle$ forms a cyclic group or subgroup of $\mathbb{Z}_p^*$, . The order of this group is the number of elements in it, and so for $\langle 6 \rangle = \mathbb{Z}_{11}^*$, we have the order $|\langle 6 \rangle| = 10$. By contrast, $\langle 3 \rangle$ forms a cyclic subgroup of $\mathbb{Z}_{11}^*$ of order 5.

Key AGREEMENT protocols

The scenario, you should already know

Alice and Bob want to establish a common secret key over a public channel. Eve can listen to the channel and she tries to obtain the secret key. (Eve can only listen but cannot modify the messages, i.e. only confidentiality is compromised but not integrity. There is also no concern on authenticity).

A Key agreement protocol between Alice and Bob establishes a common secret k. Subsequently, Alice and Bob can secure the channel using cryptographic means with k as the shared secret key. (For example, using AES with k as key to encrypt. The k is only used in this session and would be different in another session. So k is also often called the Session Key.)

$\text{Alice} \rightarrow \text{Bob}$**:** I have something to tell you but Eve can tap into our conversations. Let's first establish a common key k.

$\text{Alice} \leftrightarrow \text{Bob}$**:** Key agreement protocol to establish k.

$\text{Alice} \leftrightarrow \text{Bob}$**:** All subsequent conversations will be protected by k.

Scheme #1: the Key Agreement protocol Σ

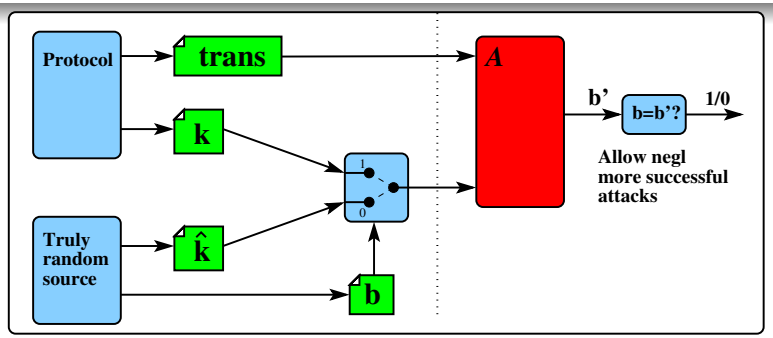
In a similar fashion to our definitions of crypto schemes/systems Π so far, we will define a protocol Σ. The difference is in interpretation of the algorithms that define the system. These algorithms operate over all participants in the scheme, and construct, or agree on, a key. It is not really an exchange, as in KEM. To emphasize this, I deviate from the book, calling this KA (not KE).

Definition 10.0: Key Agreement $\mathrm{KA} = (\mathrm{Init}, \mathrm{DoProt})$

$\mathrm{Init}(1^n, \overline{p})$**:** On input 1^n, initialize each of the p participants, with (secret) initial keys of size n.

$\mathrm{DoProt}(\overline{p})$**:** (Run the protocol) Each of the p participants run the protocol, returning a key k and a complete transcript t. We will write this as $\langle k, t \rangle \leftarrow \mathrm{DoProt}(p_i)$.

Key Agreement (KA) security as a game



The game: $\mathrm{SecKA}^{\mathrm{eav}}_{\mathcal{A},\Sigma}(n)$

1. Two parties interact using the KA protocol Σ, recorded in the transcript `trans`. Each of the parties discovers an identical key k.
2. Choose a uniform bit b. If $b = 0$, set $k' = k$, else set $k' = \{0,1\}^*$.
3. Adversary on input `trans` and k', outputs a bit b'. It wins if $b' = b$.

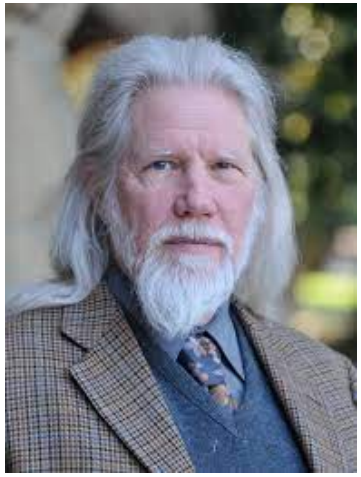
Definition 10.1 - eav-secure

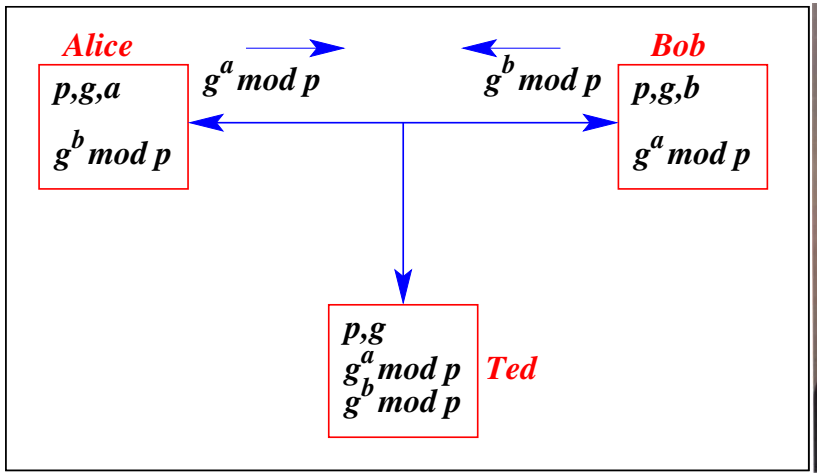
A KA Σ is secure in the presence of an eav if for all PPT $\mathcal{A}$, there is a negl, s.t.

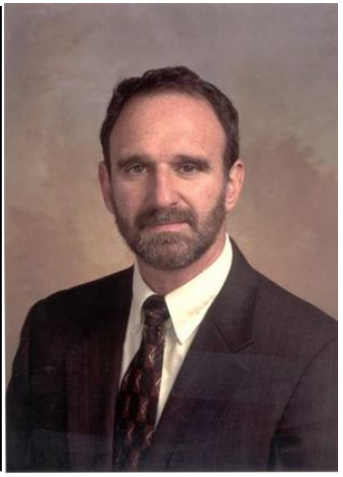
$$\Pr[\mathrm{SecKA}^{\mathrm{eav}}_{\mathcal{A},\Sigma}(n) = 1] \leq \frac{1}{2} + \mathrm{negl}(n)$$

Note that this is a much more stringent property than finding the key. This definition implies that you cannot even distinguish it from a random value.

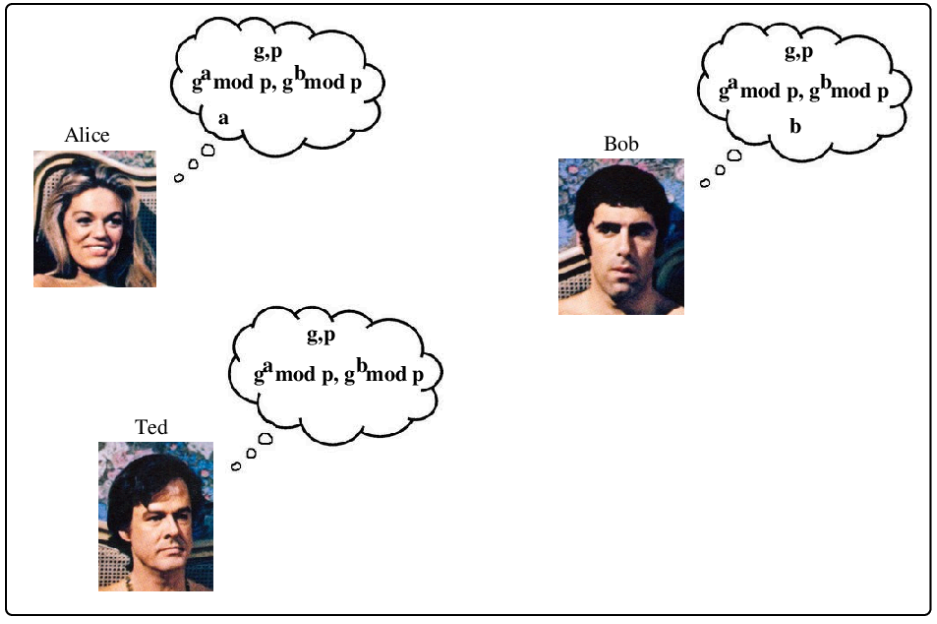
KA #1: Diffie (Whitfield), and (Martin) Hellman







After the protocol, knowledge is different...



Construction #1: Diffie-Hellman (CDH) KA

Concrete: what do the parties do? (modulo math)

Init(): Alice and Bob pre-determine a choice of p, and a generator g.
DoProt():

1. Alice, Bob choose uniform values a, b from $\mathbb{Z}_p^*$ and exchange $h_A = g^a \bmod p$, $h_B = g^b \bmod p$ ($h_A \longrightarrow$ Bob and $h_B \longrightarrow$ Alice).
2. Alice computes the secret $k_A = (g^b \bmod p)^a \bmod p = (g^b)^a \bmod p$.
3. Bob computes the secret $k_B = (g^a \bmod p)^b \bmod p = (g^a)^b \bmod p$.

Shared key is $k_A = k_B = (g^b)^a \bmod p = (g^a)^b \bmod p = g^{ab} \bmod p$.

Abstract: what do the parties do? (cyclic group)

Init(): Alice and Bob pre-determine the choice of group $\mathcal{G}$ and a generator g.
DoProt():

1. Alice, Bob choose uniform values a, b from $\mathcal{G}$ and exchange $h_a = g^a$, $h_b = g^b$ ($h_a \longrightarrow$ Bob and $h_b \longrightarrow$ Alice).
2. Alice computes the secret $k_A = h_b^a$.
3. Bob computes the secret $k_B = h_a^b$.

Note that $k_A = k_B = g^{ab}$. A common choice of group (as above) is $\mathcal{G} = \mathbb{Z}_p^*$.

Why is DH not eav-secure with $\mathbb{Z}_p^*$?

Consider definition 10.1 - eav-security

An adversary wins if it distinguishes between a random key $k' = g^z$ and the real key g^{ab}. It has the transcript **`trans`**, which in DH contains g^a, g^b, g and p.

This is exactly the DDH problem, distinguishing between $A = (g^a, g^b, g^z)$, and $B = (g^a, g^b, g^{ab})$. In the group $\mathbb{Z}_p^*$, DDH is easy to decide using the Legendre symbol. If a value such as g^a has a QR, then a must be even. We can work out the Legendre symbol for each of A or B above. For A, $\left(\frac{g^z}{p}\right)$ will be random, whereas for B the only possibilities are as follows:

$\left(\frac{g^a}{p}\right)$		$\left(\frac{g^b}{p}\right)$		$\left(\frac{g^{ab}}{p}\right)$	
1	(a even)	1	(b even)	1	(ab even)
1	(a even)	-1	(b odd)	1	(ab even)
-1	(a odd)	1	(b even)	1	(ab even)
-1	(a odd)	-1	(b odd)	-1	(ab odd)

There is an easy way to fix this vulnerability. Instead of using the group $\langle g \rangle$, use the group $\langle g^2 \rangle$. All the elements in $\langle g^2 \rangle$ have a square root.

Security of (DH) KA depends on choice of group

Theorem 10.3

If DDH is hard, then DH key agreement is eav-secure.

Relationships between techniques

By now, we should be familiar with these types of relationships:

1. DL, CDH hardness are *necessary* conditions for DH key agreement to be eav-secure (CPA-secure). But there is no proof that they are *sufficient*.
2. In our security model, we assume that the adversary cannot modify the messages. In practice, adversaries can do that. Key-agreement that is secure under such a scenario is know as *authenticated* key-agreement.
3. There is a simple way to strengthen DH key-agreement to *authenticated* key-agreement. This is done by having each entity sign their message. Using DH key-agreement together with signing is known as station-to-station protocol.
4. Station-to-station has another useful property: Forward Secrecy. Not all secure authenticated key-agreement constructions achieve forward secrecy.

Brief discussion of forward secrecy

What is forward secrecy?

A snooper is unlikely to ever be able to decrypt AES256 without access to the key. However if they record everything, and then later (somehow) retrieve the key, then they could decrypt at this later time.

Standard DH does not have this problem. Each KA generates a fresh key, and if this key is later discovered, it is only of use for that one session. By contrast, RSA does not have forward secrecy

Key EXCHANGE using PKC encryption

Eav-secure Key Exchange mechanism (KEM) from PKC

If we have an eav-secure PKC encryption, we can derive a secure key exchange. (note: is textbook RSA sufficient?)

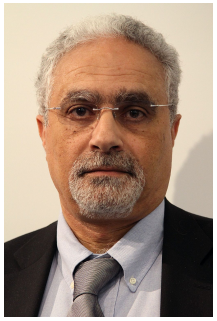
1. Alice generates (public key, private key) and sends public key to Bob.
2. Bob uniformly chooses a key k, encrypts it with Alice's public key, and sends to Alice.
3. Alice decrypts, and obtains k.

As for DH key agreement, there are ways to enhance this to an authenticated key-exchange.

However, there are some limitations.

1. k is chosen solely by Bob.
2. Generation of (public key, private key) usually is computation intensive. Hence, Alice may use a fixed (public, private) key for all exchanges. This contradicts forward secrecy.

Elgamal (Taher Elgamal - the "father" of SSL)



Protocol	Published	Status
SSL 1.0	Unpublished	Unpublished
SSL 2.0	1995	Deprecated in 2011 (RFC 6176)
SSL 3.0	1996	Deprecated in 2015 (RFC 7568)
TLS 1.0	1999	Deprecated in 2020 (RFC 8996)[8][9][10]
TLS 1.1	2006	Deprecated in 2020 (RFC 8996)[8][9][10]
TLS 1.2	2008	
TLS 1.3	2018	

Construction #2: Elgamal ENC_{p_k} ($\times$ cyclic group)

We just saw a KEM made from PKC. From KEM we can also build PKC:

$\mathrm{Gen}(1^n)$: Create secret and public keys as follows:

$s_k = \langle \mathcal{G}, g, x \rangle$ with x chosen uniformly from $\mathbb{Z}_q^*$
$p_k = \langle \mathcal{G}, g, h \rangle$ with $h = g^x$, and g a generator of $\mathcal{G}$ of order q.

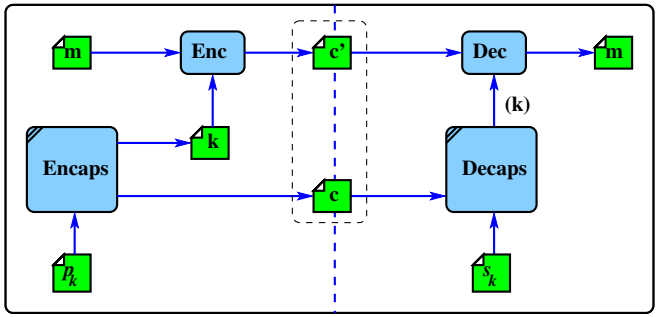
$\mathrm{Enc}_{p_k}(m)$: Choose k uniformly from $\mathbb{Z}_q^*$, output: $\langle c_1, c_2 \rangle \leftarrow \langle m \times h^k, g^k \rangle$

$\mathrm{Dec}_{s_k}(\langle c_1, c_2 \rangle)$: Output: $m \leftarrow c_1 \times c_2^{-x}$

Correctness

$$\mathrm{Dec}(\mathrm{Enc}(m)) = c_1 \times c_2^{-x} = m \times h^k \times g^{-kx} = m \times g^{kx} \times g^{-kx} = m$$

Similarity with DH key agreement



Theorem 11.18

If DDH is hard w.r.t the underyling group, then Elgamal is CPA-secure.

Elgamal follows the style of Construction 11.10 that has a Key Encapsulation Mechanism. So, we can also replace $m \times h^k$ by any other CPA-secure private-key encryption (e.g. AES in CTR mode).

Why is it so interesting? The (math) basis is strong...

1. Discrete Log problem is a hard problem in many cyclic groups. Some of the groups have useful properties, such as compact representations.
2. "ECC" Elliptic Curve Cryptography generally refers to crypto based on DL (e.g. Elgamal) with Elliptic Curve as the underlying group.
3. NIST recommends 300-bit ECC key size, but >2000 bits for RSA.

Another interesting thing

Elgamal is homomorphic w.r.t. multiplication

$$(m_1 \times m_2) = \mathrm{Dec}(\mathrm{Enc}(m_1) \odot \mathrm{Enc}(m_2))$$

where $\times$ is modulo multiplication, and $\odot$ is some operation that can be efficiently carried out in the encrypted domain. (What could it be?)

Are there homomorphic encryptions w.r.t. addition? i.e.

$$(m_1 + m_2) = \mathrm{Dec}(\mathrm{Enc}(m_1) \odot \mathrm{Enc}(m_2))$$

Yes, see the Paillier cryptosystem (optional) Chapter 13.2. The security of Paillier is based on another hardness assumption in modulo N^2.

This leads to a chosen ciphertext attack

Elgamal encryption is not secure under chosen ciphertext attack.

For example:

> *Given $\langle c_1, c_2 \rangle$ of some message m (which need not be known by the attacker), you can construct $\langle c_1, 2c_2 \rangle$ of the message $2m$.*

An appropriate padding technique must be used to counter this.

Python Elgamal code

Python scripts for encoding c_1 and c_2

```
./Elgamal.py m        q        α        y
./Elgamal.py "Elgamal" 10168938831019335571 15 4721718615616538173
```

```
m     = string_to_int(sys.argv[1])    # get values from arguments
q     = int(sys.argv[2])
alpha = int(sys.argv[3])
y     = int(sys.argv[4])
k     = random.randrange(10,1000)     # Choose random k
K     = modexp(y,k,q)                 # Compute key  K = y^k mod q
c1    = modexp(alpha,k,q)             # c1 = α^k mod q
c2    = (K*m)%q                       # c2 = mK mod q
print c1,c2
```

Python scripts for decoding c_1 and c_2

```
./unElgamal.py c1       c2        x        q
./unElgamal.py 3838410137284106619   50140057099304035 75   29   10168...
```

```
c1 = int(sys.argv[1])     # get values from arguments
c2 = int(sys.argv[2])
x  = int(sys.argv[3])
q  = int(sys.argv[4])
K  = modexp(c1,x,q)       # Calculate K = c1^x mod q
Ki = modInvEuclid(K,q)    # Calculate inverse K^-1 mod q
m  = (c2*Ki)%q            # Calculate m = c2 K^-1 mod q
print int_to_string(m)
```

Keygen $\mathrm{GenECC}(1^n)$, and ECC $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$

GenECC: creates public keys p_k and s_k

1. Using an (additive) elliptic group $\mathcal{G} = E_p(a, b)$, select a generator g in the group which has a large order n. Note that in an additive group, $ng = 0$ is like $g^n = 1$ in a multiplicative group.
2. Uniformly choose $n_A : n_A < n$. s_k is $\langle \mathcal{G}, g, n_A \rangle$
3. Calculate $P_A = n_A g$. p_k is $\langle \mathcal{G}, g, P_A \rangle$

Construction #3: ENC_{p_k} using ECC (+ cyclic group)

$\mathrm{Gen}(1^n)$: Compute $(p_k, s_k) \leftarrow \mathrm{GenECC}(1^n)$.

$\mathrm{Enc}_{p_k}(m)$: Given p_k and a message $m \in \mathcal{G}$, choose k uniformly from $\mathbb{Z}_n^*$, and encrypt: $\langle c_1, c_2 \rangle \leftarrow \langle m + kP_A, kg \rangle$.

$\mathrm{Dec}_{s_k}(\langle c_1, c_2 \rangle)$: Given $s_k, \langle c_1, c_2 \rangle$, calculate: $m \leftarrow c_1 - n_A c_2$.

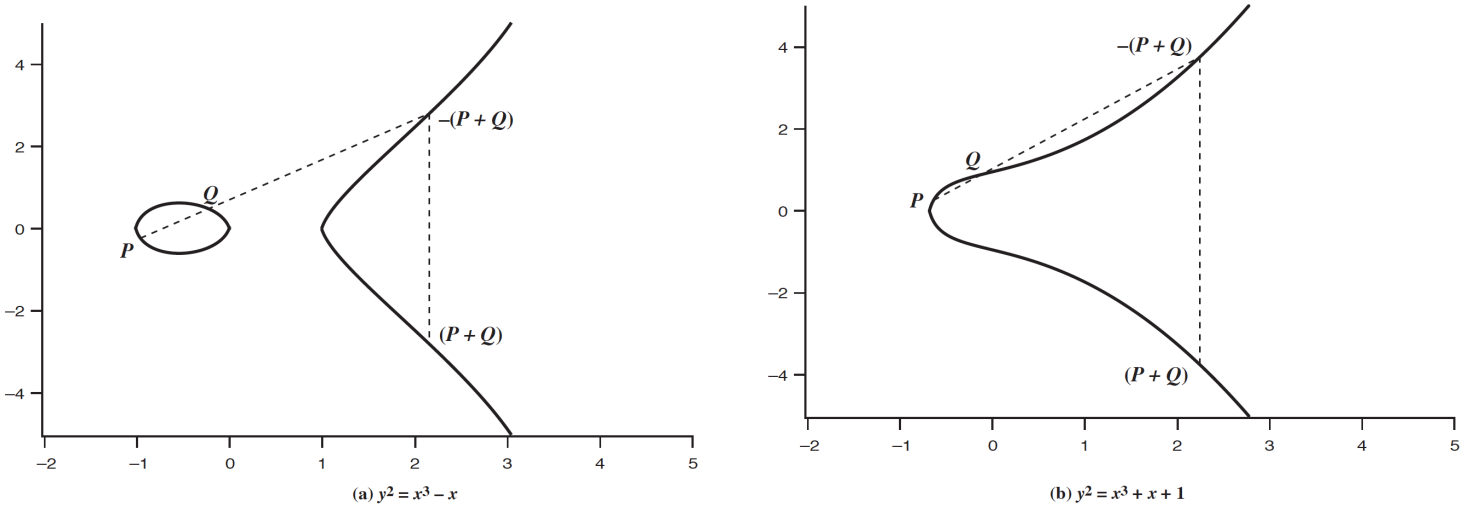
Correctness

$$\mathrm{Dec}(\mathrm{Enc}(m)) = c_1 - n_A c_2 = m + kP_A - n_A kg = m + kn_A g - n_A kg = m$$

ECC - what is the group?

Consider addition over cubic elliptic curves, such as

$y^2 = x^3 + ax + b$, with zero O, and the points $E(a, b) = \{(x_0, y_0), (x_1, y_1), \ldots\}$ on the curve. The elements are points on a plane not on a number line.



An addition operation $+_{E(a,b)}$ for points on this curve: the sum of $P +_{E(a,b)} Q$ is reflection of the intersection R. The group is $\langle +_{E(a,b)}, E(a, b) \rangle$.

ECC uses curves whose elements are finite: prime curves $E_p(a, b)$ defined over $\mathbb{Z}_p$, and binary curves $E_{2^m}(a, b)$ defined over $GF(2^m)$.

ECC adding: real and in $E_p(a, b)$

Adding points in $E(a, b)$ (Real curve)

If we had $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$, and $P \neq \pm Q$. We can find $R = P +_{E(a,b)} Q$ by finding gradient of line, and then intersection with curve:

$$\begin{aligned} \text{Gradient:} \quad & \Delta = \frac{y_Q - y_P}{x_Q - x_P} \\ x \text{ coordinate for } R\text{:} \quad & x_R = \Delta^2 - x_P - x_Q \\ y \text{ coordinate for } R\text{:} \quad & y_R = \Delta(x_P - x_R) - y_P \\ \text{Finally:} \quad & R = (x_R, y_R) \end{aligned}$$

$P +_{E(a,b)} P$ uses a different method

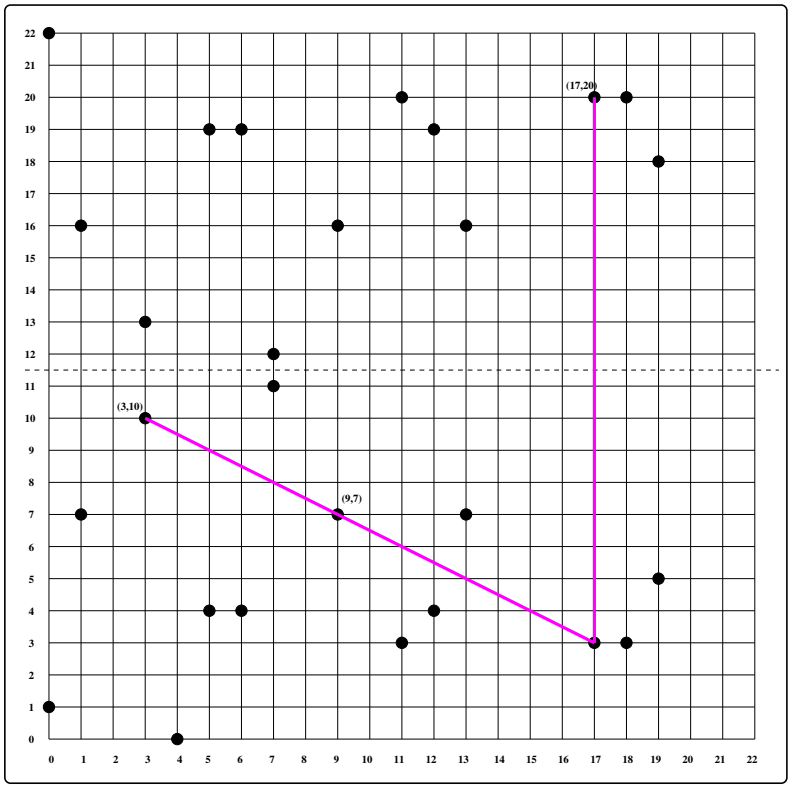
Adding points in $E_p(a, b)$ (Finite field)

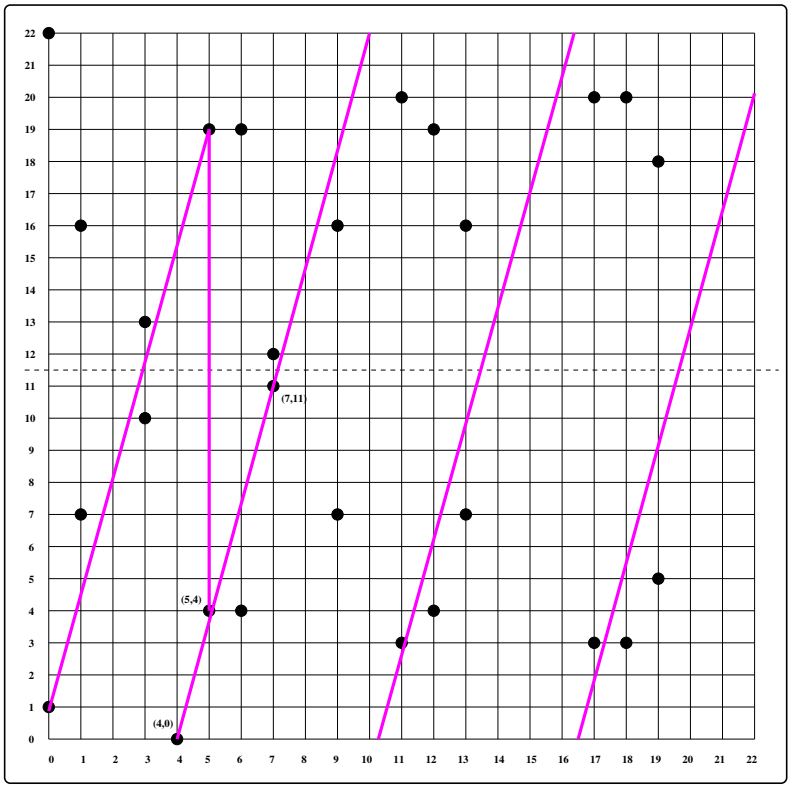
We find R as before, modulo p:

$$\begin{aligned} \text{Gradient:} \quad & \Delta = \frac{y_Q - y_P}{x_Q - x_P} \bmod p \\ x \text{ coordinate for } R\text{:} \quad & x_R = \Delta^2 - x_P - x_Q \bmod p \\ y \text{ coordinate for } R\text{:} \quad & y_R = \Delta(x_P - x_R) - y_P \bmod p \\ \text{Finally:} \quad & R = (x_R, y_R) \end{aligned}$$

$P +_{E_p(a,b)} P$ again uses the different method.

Curve $y^2 = x^3 + x + 1 \bmod 23$ does not "look" nice





Example $P +_{E_{23}(1,1)} Q$ operations

P+Q	Gradient Δ	x coordinate x_R	y coordinate y_R	R
$(3,10) + (9,7)$	$= \frac{-3}{6} = 11$	$= 11^2 - 3 - 9 = 17$	$= 11(3-17) - 10 = 20$	$(17,20)$
$(4,0) + (7,11)$	$= \frac{11}{3} = 19$	$= 19^2 - 4 - 7 = 5$	$= 19(4-5) - 0 = 4$	$(5,4)$

Why use ECC?

ECC: for signatures, encryption, key-exchange...

ECC addition is an analog of modulo multiply. ECC repeated addition is an analog of modulo exponentiation.

We have a "hard" problem, equivalent to the discrete log problem. Consider $Q = kP$, where Q, P belong to a prime curve... it is

- "easy" to compute Q given k, P but
- "hard" to find k given Q, P.

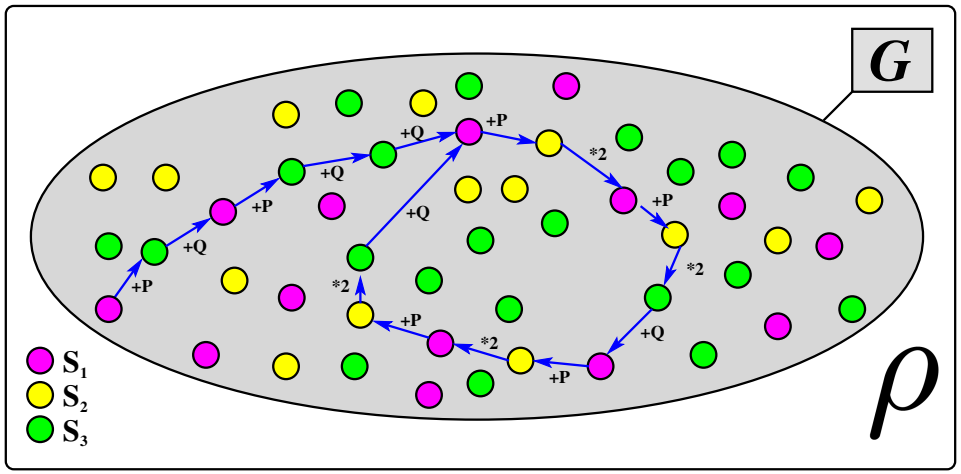
This is known as the elliptic curve logarithm problem.

Comparable key sizes (in bits)

Security	Symmetric	FFC: DSA	IFC: RSA	ECC,DH
128	AES-128	3072/256	3072	256
192	AES-192	7680/384	7680	384
256	AES-256	15360/512	15360	512

https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf

Best attack on ECC so far is Pollard-rho/Floyd!



Want to find $k : kP = Q$? Cast it as a collision!

The world is a chain of elements of the group, partitioned into three (S_1, S_2, S_3) types (coloured). We either double the element, or add P or Q:

$$P_{i+1} = f(P_i) \quad = \quad \begin{cases} P_i + P & \text{if} \quad P_i \in S_1 \\ 2P_i & \text{if} \quad P_i \in S_2 \\ P_i + Q & \text{if} \quad P_i \in S_3 \end{cases}$$

Each P_i is some sum $aP + bQ$. On collision, $a_1 P + b_1 Q = a_2 P + b_2 Q$, and

$$\frac{a_1 - a_2}{b_2 - b_1} P \quad = \quad Q$$

Sidebar: Calculating $2P = P +_{E_P(a,b)} P$ for ECC

Doubling a point in $\langle +_{E_p(a,b)}, E_p(a,b) \rangle$

- If $y_P = 0$, return O, the zero point.
- $P = (x_P, y_P)$, and $y_P \neq 0$.
- Find $R = P +_{E_p(a,b)} P$ by finding gradient of the tangent, and then intersection with curve:

Gradient:	$\Delta = \frac{3x_P^2 + a}{2y_P} \bmod p$
x coordinate for R:	$x_R = \Delta^2 - 2x_P \bmod p$
y coordinate for R:	$y_R = \Delta(x_P - x_R) - y_P \bmod p$
Finally:	$R = (x_R, y_R)$

All operations modulo 23...

P+P	Gradient Δ	x coordinate x_R	y coordinate y_R	R
$(7,11) + (7,11)$	$= \frac{10}{22} = 13$	$= 13^2 - 14 = 17$	$= 13(7 - 17) - 11 = 20$	$(17, 20)$
$(9,7) + (9,7)$	$= \frac{14}{14} = 1$	$= 1^2 - 18 = 6$	$= 1(9 - 6) - 7 = 19$	$(6, 19)$

Sidebar: On $\frac{a}{b} \bmod p$ calculation for ECC

Dividing by b is the same as multiplication by b^{-1}:

To calculate $\frac{a}{b} \bmod p$,

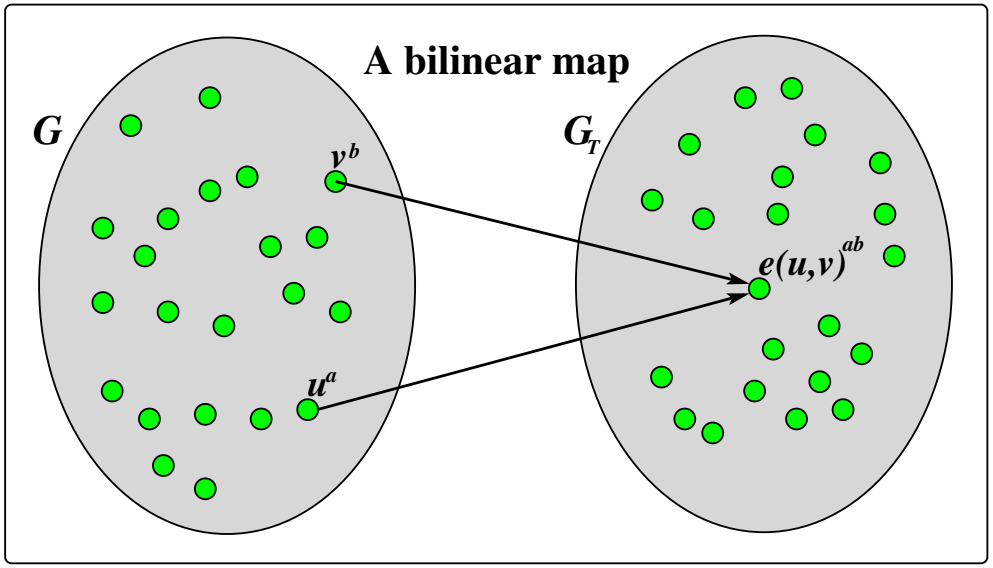
1. Use extended euclidean algorithm to calculate $b^{-1} \bmod p$.
2. Then multiply $a \times b^{-1} \bmod p$.

All operations modulo 23...

To calculate $\frac{7}{11} \bmod 23$

1. Calculate $11^{-1} \bmod 23 = 21$.
2. $7 \times 21 \bmod 23 = 9$

Bilinear maps/pairings



Pairings...

A (symmetric style) bilinear map is a pairing between two elements of a group to a second (same order) group: $G \times G \rightarrow G_T$. There are also (asymmetric) pairings like $G_1 \times G_2 \rightarrow G_T$.

Such a symmetric mapping (if it can be efficiently computed) can be used to solve DDH: $z = ab$ iff $e(g, g^z) = e(g^a, g^b)$. Originally this was the focus of bilinear maps in crypto, and this is why CDH is considered *harder* than DDH.

The other thing is that it can allow you to reduce the discrete log problem in G to a (perhaps simpler) discrete log problem in G_T: $e(g, g^a) = e(g, g)^a$.

A little more on bilinear maps/pairings

Can we have “simple” pairings?

This is easy, but unfortunately such "simple" maps are not useful for crypto. A simple map based on (say) integers might be the additive integer group $G = \langle \mathbb{Z}, + \rangle$, and $G_T = \langle \mathbb{Z}, * \rangle$, with

$$e(u, v) = 2^{uv}$$

For the additive group, u^2 means $u + u$, and it is sufficient to show that $e(u + u, v) = e(u, v)^2$ and $e(u, v + v) = e(u, v)^2$. Lets try an example:

$$e(3, 4) = 2^{12}$$

and

$$\begin{aligned} e(3 + 3, 4) &= e(3, 4)^2 \\ &= e(3, 4) \times e(3, 4) \\ &= 2^{12} \times 2^{12} \\ &= 2^{24} \\ &= e(3, 4 + 4) \end{aligned}$$

A little more on bilinear maps/pairings

$G \rightarrow$

+	a^0	a^1	a^2	a^3	a^4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

$G_T \rightarrow$

a	a^1	a^2	a^3	a^4	a^5	a^6	a^7
2	2	4	8	5	10	9	7
3	3	9	5	4	1	3	9
4	4	5	9	3	1	4	5
5	5	3	4	9	1	5	3
6	6	3	7	9	10	5	8

"Simple" pairings with finite groups?

The groups $G = \mathbb{Z}_5^+$ and $G_T = \langle 3 \rangle_{11}$, with $e(u, v) = 3^{uv} (\mathrm{mod} 11)$:

$$\begin{aligned} e(4,3) &= 3^{12} \bmod 11 \\ &= 9 \end{aligned}$$

and

$$\begin{aligned} e(4+4,3) &= e(4,3)^2 \\ &= e(4,3) \times e(4,3) \\ &= 3^{24} \bmod 11 \\ &= 4 \\ &= e(4,3+3) \end{aligned}$$

This sort of map is not too useful because the groups involving simple integer style math tend to be invertible - knowing u and $e(u, v)$ you can determine v.

The big picture..

In practice? G is an elliptic curve, G_T is a finite field...

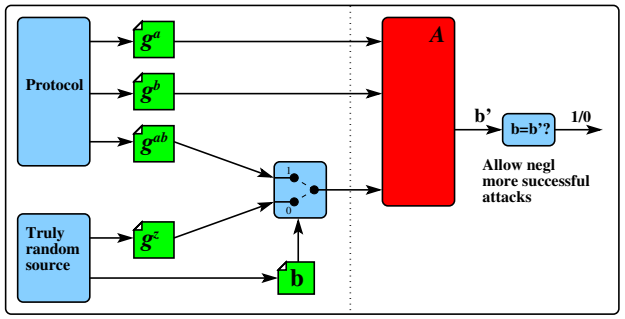
There are getting to be more uses of pairing based crypto, including IBE (this week), simple encryption, signatures (next week) and so on. Note that

$$e(u^a, v^b) = e(u^b, v^a)$$

(one for encryption, one for decryption).

In a talk by Boneh, the above diagram showed crypto history, from one-way functions, discrete logs, pairings, LWE (Learning with Errors), through to multilinear maps for indistinguishability obfuscation. What comes next?

Formal DDH game and definition: $\mathrm{Adv}^{\mathrm{DDH}}_{\mathcal{A},\mathcal{G}}$



DDH game definition elements

Assuming a fixed g, a generator of a cyclic group $\mathcal{G}$ of order q, the challenger randomly picks a 3-tuple (a, b, z) of elements from $\mathcal{G}$, and a bit b. Then according to the bit b, the challenger outputs either $c_0 = (g, g^a, g^b, g^z)$ or $c_1 = (g, g^a, g^b, g^{ab})$. The adversary outputs a bit b' and succeeds if $\mathrm{b}' = \mathrm{b}$, with output 1. We can define the advantage of the adversary solving as

$$\mathrm{Adv}^{\mathrm{DDH}}_{\mathcal{A},\mathcal{G}} = |\Pr[\mathcal{A}(c_0) = 1] - \Pr[\mathcal{A}(c_1) = 1]|$$

Definition 8.63 - DDH-hard

The DDH problem is hard relative to $\mathcal{G}$ if, for any PPT adversary $\mathcal{A}$, there is a negl, s.t.

$$\Pr[\mathrm{Adv}^{\mathrm{DDH}}_{\mathcal{A},\mathcal{G}}] \leq \mathrm{negl}(n)$$

Bilinear maps more formally

Bilinear map definition

Our game/definition 10.1 for eav-secure for Diffie-Hellman KA corresponds to solving the DDH problem[a]. We have already seen that DDH is not hard in the multiplicative group $\mathbb{Z}_p^*$ (using the Legendre symbol). DDH is also not hard in some elliptic curves[b] over $\mathrm{GF}(p)$ (where bilinear maps can be used).

Bilinear map: Let $G = \langle g \rangle$ and G_T be two (same order) groups. A bilinear map is a *pairing* function $e : G \times G \to G_T$ s.t. $e(u^a, v^b) = e(u, v)^{ab}$ for all u, v in G and all integers a, b. In particular, if $h = e(g, g)$, then $e(g^a, g^b) = e(g, g)^{ab} = h^{ab}$.

The bilinear map can be used to break DDH, with ppt algorithms to compute e: `https://people.csail.mit.edu/alinush/6.857-spring-2015/papers/bilinear-maps.pdf`

Some properties of bilinear maps turn out to be useful in constructing new primitives. In particular we have 3-party KA, and an identity based encryption (IBE) construction.

---

[a] But not the CDH problem - at this stage CDH is still believed to be hard.

[b] Coming up soon...

3-party key agreement protocol

Construction #4: 3-party KA using bilinear map

Alice, Bob, Charles want to establish a common key over an unsecured broadcast channel where the adversary can eavesdrop but not modify the data in the channel. We could use DH KA, extended to three parties, but this would require extra communications between the parties.

An efficient protocol can make use of the bilinear map. We assume that everyone has agreed on a choice of bilinear map e and a generator g with order q. The protocol has two steps, similar to DH KA. We will have:
DoProt():

1. Broadcast secrets

   Alice randomly picks an integer a in $\mathbb{Z}_q^*$ and broadcasts g^a.

   Bob randomly picks an integer b in $\mathbb{Z}_q^*$ and broadcasts g^b.
   Charles randomly picks an integer c in $\mathbb{Z}_q^*$ and broadcasts g^c.

2. Compute keys (noting that $h = e(g, g)$):

   Alice computes: $k = e(g^c, g^b)^a = h^{abc}$

   Bob computes: $k = e(g^c, g^a)^b = h^{abc}$

   Charles computes: $k = e(g^a, g^b)^c = h^{abc}$

Scheme #2: Identity based encryption

Can we avoid PKI? Identity based encryption (IBE)?

It would be convenient if the name of an entity is its public key.

- Proposed by Shamir in 1984. However, construction of such a scheme remained an open problem for many years.
- Boneh/Franklin gave the first practical IBE based on *pairing* in 2002.

The scheme $\mathrm{IBE} = (\mathrm{Setup}, \mathrm{Enc}, \mathrm{Dec})$

We assume a central Trusted Authority (TA) with a master private key M_s, and public key M_p, securely distributed to each user. Each user has a name nm, perhaps their email address. Every user knows each other's names.

$\mathrm{Setup}(1^n)$**:** For each user U, TA generates the user private key k_U, and securely sends it to the user.

$\mathrm{Enc}_{\mathrm{nm},M_p}(m)$**:** Bob encrypts a message: $c \leftarrow \mathrm{Enc}_{\mathrm{nm},M_p}(m)$

$\mathrm{Dec}_{k_U,M_p}(c)$**:** Alice decrypts: $m \leftarrow \mathrm{Dec}_{k_U,M_p}(c)$

There is no PKI in this system. Bob doesn't need Alice's certificate. He simply takes her name (alice@comp.nus.edu.sg) to derive the public key.

IBE remarks and a construction

Limitations of the scheme

1. It requires a trusted authority to generate all private keys. Since the TA knows the private keys, it can "forge" the signature of any user.
2. It is difficult to revoke a key.

Construction #5: Boneh/Franklin IBE using bilinear maps

$\text{Setup}(1^n)$: Find a suitable bilinear map e with $\mathcal{G}, \mathcal{G}_T$, where $\mathcal{G}$ has order q. Let g be a generator of $\mathcal{G}$, and $h = e(g, g)$. Choose two suitable mappings $\mathcal{H}_1, \mathcal{H}_2$, where the range of $\mathcal{H}_1$ is $\mathcal{G}$, and the domain of $\mathcal{H}_2$ is $\mathcal{G}_T$. Uniformly pick a number $s \in \mathbb{Z}_q^*$ as the Master private key. Set the master public key $M_p = g^s$. Given a name $U = \text{nm}$, the public key is $k = \mathcal{H}_1(\text{nm})$ and the matching private key is $k_U = \mathcal{H}_1(\text{nm})^s$.

$\text{Enc}_{\text{nm},M_p}(m)$: Randomly choose $r \in \mathbb{Z}_q$, compute $t := e(\mathcal{H}_1(\text{nm}), M_p)$, and then output: $\langle c_1, c_2 \rangle \leftarrow \langle m \oplus \mathcal{H}_2(t^r), g^r \rangle$

$\text{Dec}_{k_U,M_p}(\langle c_1, c_2 \rangle)$: Decrypt this way: $m \leftarrow c_1 \oplus \mathcal{H}_2(e(k_U, c_2))$

Discussion

Correctness of the construction

Note that $m \oplus \mathcal{H}_2(t^r)$ can be viewed as an encryption of m using $\mathcal{H}_2(t^r)$ as the key. So, as long as $\mathcal{H}_2(t^r)$ can be recovered, the message m can be obtained. i.e. If t^r can be recovered, the message m can be obtained.

As before, given a name $U = \mathrm{nm}$, the public key is $k = \mathcal{H}_1(\mathrm{nm})$, and the matching private key is $k_U = \mathcal{H}_1(\mathrm{nm})^s = k^s$. The recipient of the encrypted message has $k_U, \langle c_1, c_2 \rangle$ and knows the fixed parts of the scheme e, M_p and so on. The recipient can compute

$$e(k_U, c_2) = e(k^s, g^r) = e(k, g)^{rs}$$

but notice that the sender used

$$t^r = e(k, M_p)^r = e(k, g^s)^r = e(k, g)^{rs}$$

It is magic! To get an intuition about how it works, think of IBE as an extension of 3-party DH key exchange.

Symmetric encryption schemes

$$\mathrm{ENC}_k = (\mathrm{Gen}(1^n), \mathrm{Enc}_k(m), \mathrm{Dec}_k(c)) \qquad \textbf{(ch1)}$$

Properties	Defs	Constructions	Proofs
Perfect secrecy	2.3	(One time pad)	Thm 2.9
↕			Lemma 2.6
Perfect indistinguishability	2.5		
↓			
EAV-secure	3.8	3.17 (PRG)	Thm 3.18
↑			Proof given
EAV-Multi-encryption	3.19		
↑			
CPA-Multi-encryption	3.22		
↕			Thm 3.24
CPA-secure	3.23	3.30 (PRF)	Thm 3.31
↑			
CCA-secure	3.33		
+			
Unforgeable	4.16		
↕			Thm 4.19
Authenticated	4.17	4.18 (Enc-then-Auth)	

MAC and HASH schemes

$$\mathrm{MAC}_k = (\mathrm{Gen}(1^n), \mathrm{Mac}_k(m), \mathrm{Vrfy}_k(m,t)) \quad \textbf{(4.1)}$$

Properties	Defs	Constructions	Proofs
Unforgeable	4.2		
↑			Discussed
Strong	4.3	4.5 (Using PRF)	Thm 4.5
		4.7 (Variable length)	Thm 4.8
		4.11 (CBC-MAC)	Thm 4.12
		4.11(a) (Variable CBC-MAC)	

$$\mathrm{HASH} = (\mathrm{Gen}(1^n), \mathcal{H}^s(x)) \quad \textbf{(5.1)}$$

Properties	Defs	Constructions	Proofs
CollisionResistant	5.2	5.3 (Long message)	

Asymmetric encryption schemes

$$\mathrm{ENC}_{p_k} = (\mathrm{Gen}(1^n), \mathrm{Enc}_{p_k}(m), \mathrm{Dec}_{s_k}(c)) \quad \textbf{(11.1)}$$

Properties	Defs	Constructions	Proofs
		11.26 (Textbook RSA) (Elgamal) (ECC)	Thm 11.18
DDH-hard Factor-hard ↑ RSA-hard	8.63 8.45 8.46		
EAV-secure ↕ CPA-secure ↑ CCA-secure	11.2	11.32 (1-Bit RSA)	

Commitment, KEM, IBE schemes

$$\mathrm{COM} = (\mathrm{Setup}(1^n), \mathrm{Commit}(a), \mathrm{Open}(c_a)) \quad \textbf{(ch5)}$$

Properties	Defs	Constructions	Proofs
Secure (Binding+Hiding)	5.13	(Hash based system)	

$$\mathrm{KEM}_{p_k} = (\mathrm{Gen}(1^n), \mathrm{Encaps}_{p_k}(1^n), \mathrm{Decaps}_{s_k}(c)) \quad \textbf{(11.9)}$$

Properties	Defs	Constructions	Proofs
CPA-secure		11.10 (CPA_KEM+EAV_ENC)	Thm 11.12
CCA-secure	11.13	11.10 (CCA_KEM+CCA_ENC)	Thm 11.13
		11.37 (CCA_KEM+TextbookRSA)	

$$\mathrm{IBE}_{p_k} = (\mathrm{Setup}(1^n), \mathrm{Enc}_{\mathrm{nm},M_P}(m), \mathrm{Dec}_{k_U,M_P}(c)) \quad \textbf{(...)}$$

Properties	Defs	Constructions	Proofs
		(Boneh/Franklin)	

KA scheme

$$\mathrm{KA} = (\mathrm{Init}(1^n, \overline{p}), \mathrm{DoProt}(\overline{p}))$$ **(...)**

Properties	Defs	Constructions	Proofs
KA-EAV-secure	10.1	(DHKE) 3-party Bilinear maps	Thm 10.3