Electronic Voting Seminar / Tomer Levinboim

October 20, 2009

Content

- DLOG
- Encryption Algorithms ElGamal & Paillier
- Threshold encryption (using ElGamal)
- Zero Knowledge Proofs and non interactive ZKP
- Voting System: Prêt à Voter
- Appendix: Math Reminder
- References

Discrete Log Problem

Let g be a generator of some cyclic group G and let $h \in G$. The Discrete Log Problem (DLOG) asks us to find a solution x for the equation $g^x = h$.
In the rest of this document, we shall regard G as a subgroup of $\mathbb{Z}_p^*$ of order q for primes p, q. Under this context the DLOG problem asks to find a solution for

$$g^x = h \bmod p$$

Where $x \in \mathbb{Z}_q$.
Currently, no efficient (polynomial time) classical algorithm for computing general discrete logarithms is known (see Math Appendix, Chinese Remainder Theorem subsection for a short discussion).

Encryption

We shall use two different public-key encryption systems, ElGamal and Paillier, named after their inventors. These cryptosystems share the following properties:

- Probabilistic - In a probabilistic cryptosystem, any specific plaintext has many ciphertexts (but not the other way around). This property is advantageous when one retransmits a message but wishes to conceal the fact that it is the same message.
- Homomorphic - Homomorphic encryption is a form of encryption where one can perform a specific algebraic operation on the plaintext by performing a (possibly different) algebraic operation on the ciphertext. For example, taking multiplying two ciphertexts might correspond to adding the plaintexts, as is the case with Paillier encryption. See equations (1) and (3) below.

ElGamal

The following sections describes how to key generation, encryption and decryption are performed according to ElGamal.

Key Generation

- Alice generates an efficient description of a multiplicative cyclic group G, of order q with generator g.
- Alice chooses a random $x \in Z_q$
- Alice computes $h := g^x$
- Alice publishes h, along with the description of G, q, g, as her public key. Alice retains x as her secret private key

Encryption

To send a message m to Alice using her public key (G, q, g, h):

- Bob chooses $r \in Z_q$ at random
- Bob computes the ciphertext $c := (c_1, c_2) := (g^r, m \cdot h^r)$ and sends to Alice

Decryption

To decrypt (c_1, c_2):

- Alice computes $s := c_1^x = g^{xr} = h^r$
- Alice retrieves the message by computing $c_2 \cdot s^{-1} = m \cdot h^r \cdot s^{-1} = m$

Homomorphism

Note that:

$$(1)\ E(m_1 \cdot m_2, r_1 + r_2) = (g^{r_1+r_2},\ m_1 m_2 h^{r_1+r_2})$$

$$= (g^{r_1}, m_1 h^{r_1}) \cdot (g^{r_2}, m_2 h^{r_2}) = E(m_1, r_1) \cdot E(m_2, r_2)$$

Meaning, ElGamal is multiplicative homomorphic.
In particular, taking $m_2 = 1$ (the identity element of the cyclic group G) we obtain a new valid encryption of m_1, or rather a *re-encryption*:

$$(2)\ E(m_1, r_1) \cdot E(1, r_2) = (g^{r_1+r_2},\ m_1 h^{r_1+r_2}) = E(m_1,\ r_1 + r_2)$$

Remarks

1. We shall work in a cyclic group $G \subset Z_p^*$ of order $q = \frac{p-1}{2}$ for some prime p, that is q is a Sophie Germain[1] prime (The relation between Sophie Germain primes and hardness of DLOG is explained in the appendix). Note that the multiplication operator in G is the same one we use in $\mathbb{Z}_p^*$, that is multiplication mod p (and not q).

2. Given an efficient algorithm A for solving DLOG, the eavesdropper Eve could effectively decrypt an ElGamal ciphertext $(c_1, c_2) = (g^r, m \cdot h^r)$ by simply using A to factor out the private-key x from $h = g^x$ and proceeding to decrypt as usual (similarly, Eve could find the DLOG of $c_1$and use it to calculate $h^r = g^{xr}$ instead).

Paillier

TBD

Threshold Encryption (using ElGamal)

Public-key cryptosystems are designed such that the private-key is held by a single entity, and only it may decrypt a ciphertext that was created with the corresponding public-key. However, sometimes it is useful to distribute the private key among several parties (say n) such that any subset of more than $t \leq n$ parties could decrypt the ciphertext. Such a system is called a (t, n)-threshold cryptosystem.

Denote the n parties by $A_1, ..., A_n$ and let p, q be prime numbers such that $q|p-1$ (e.g. q is a Sophie-Germain prime, $p = 2q + 1$), let $g \in \mathbb{Z}_p^*$ of order q.

[1] It is conjectured that there are infinitely many Sophie Germain primes, but this has not been proven (wiki).

Key Generation

1. Each party A_i selected a **secret private** value v_i and randomly selects a polynomial $f_i()$ of degree t, such that $f_i(0) = v_i$

$$f_i(x) = v_i + a_{i,1}x + ... + a_{i,t}x^t$$

   (a) Let $f(x) = \sum_{i=1}^{n} f_i(x)$ be the general polynomial

2. Each A_i then publishes g^{v_i}, $g^{a_{i,1}}$,...,$g^{a_{i,t}}$ (a commitment to the coefficients of his polynomial), where g^{v_i} is regarded as $A_i's$ public key.

3. Each A_i sends to each A_j (using A_j's public key g^{v_j} or a private secure channel) the value of his polynomial at j - $f_i(j)$, denote this message by $u_{i,j}$

4. At this point, each A_j can determine if the message he received matches the commitment

   (a) A_j doesn't know how to directly check $u_{i,j} = f_i(j)$ since the coefficients of f_i are unknown to him.

   (b) However, he can check whether

$$g^{u_{i,j}} = g^{v_i} \cdot (g^{a_{i,1}})^j \cdot ... \cdot (g^{a_{i,t}})^{j^t} = g^{v_i + a_{i,1}j + ... + a_{i,t}j^t} = g^{f_i(j)}$$

   (c) After receiving and validating$u_{1,j}, ..., u_{n,j}$ A_j computes $s_j = \sum_{i=1}^{n} f_i(j) = f(j)$ and keeps it **private**. Also let $h_j = g^{s_j}$ which A_j makes **public**.

5. Let $s = \sum v_i$ be the private-key (which no one can compute). The system's public key is $y = \prod g^{v_i} = g^s = g^{\sum v_i} = g^{\sum f_i(0)} = g^{f(0)}$ (which anyone can compute via $\prod g^{v_i}$, since all g^{v_i} are public).

To sum up, the private-key s is unknown to all, the public key y is known to all, v_i is known only to A_i and $s_j = f(j)$ is known only to A_j.
So each A_j knows exactly one point on the polynomial f - the point $(j, f(j))$, and cannot compute others.

Encryption

To encrypt a message m we use the aforementioned ElGamal cryptosystem as usual to obtain a ciphertext (c, d).

Decryption

To decrypt (c, d) we proceed as follows:

1. A_j publishes $w_j = c^{s_j}$ and provides a ZKP for equality of DLOG between w_j and his earlier public commitment h_j

2. Given a set A of $t+1$ parties that passed the above test, decrypting is done with Lagrange interpolation (see reminder in appendix) as follows:
   (a) Evaluate $\lambda_{j,A}(x) = \prod_{i \in A/\{j\}} \frac{x-i}{j-i} \bmod q$, at $x = 0$ (that is $\lambda_{j,A}(0) = \prod_{i \in A/\{j\}} \frac{i}{i-j} \bmod q$). Here, operations are taken over $\mathbb{F}_q$.
   (b) Compute $c^s = c^{f(0)} = c^{\sum \lambda_{j,A}(0) s_j \bmod q} = \prod w_j^{\lambda_{j,A}(0)}$ where multiplication is performed over $\mathbb{F}_p$.

Each of the parties can compute c^s, however, s remains a secret (as it should), and decrypting is done as usual for ElGamal:

$$m = d(c^s)^{-1}$$

When implementing, take special notice if the operations are done in $\mathbb{Z}_p^*$ and in $\mathbb{Z}_q^*$. If you know your Number Theory, it should be easy.

Zero Knowledge Proofs (ZKP)

In cryptography, a zero-knowledge proof or zero-knowledge protocol is an interactive method for one party to prove to another that a (usually mathematical) statement is true, without revealing anything other than the veracity of the statement (wiki).

Equality of DLOG

Let $g, h \in Z_p^*$ and let $(a,\ b) = (g^{x_1} \bmod p,\ h^{x_2} \bmod p)$ be public (while x_1, x_2 are not). Suppose the prover P knows the discrete logs of a and b (that is x_1, x_2) and claims they are the same, i.e., $log_g a = log_h b$. Denote $x = x_1 = x_2$ and suppose now P wished to prove this equality to a verifier V without revealing a single bit of data about x. The following protocol (Chaum-Pedersen, 92) is a ZKP for Equality of DLOG:

1. **Commitment**: P selects $r \in \mathbb{Z}_q$ uniformly at random computes $u = g^r$, $v = h^r$ and sends u, v to V
2. **Challenge**: V selects $c \in \mathbb{Z}_q$ uniformly at random and sends to P
3. P sends $z = r + cx$ to V

The verifier can now check whether $ua^c = g^z$ and $vb^c = h^z$, it accepts if so, and rejects otherwise.

At this point, three things should be considered, the *completeness* and *soundness* of the protocol and its *zero-knowledge* property (meaning, that the execution of the protocol didn't reveal anything other than the equality of DLOG, even if the verifier is dishonest).

Completeness: If P knows x, the verifier will always accept since $ua^c = g^r \cdot g^{cx} = g^z$ and $vb^c = h^r \cdot h^{cx} = h^z$.

Soundness: Suppose the statement is false, $a = g^{x_1}$ and $b = h^{x_2}$ such that $log_g(a) \neq log_h(b)$ can P still fool V?
P can only cheat with u and v, as he may pick $u = g^{r_1}$ and $v = g^{r_2}$, at which point he will "win" only if

$$r_1 + cx_1 = r_2 + cx_2$$

Meaning $r_1 - r_2 = c(x_2 - x_1)$ however, since c was chosen randomly, $c(x_2 - x_1)$ could be any element in $\mathbb{Z}_q$, so P will succeed if he chose any r_1, r_2 whose difference is $c(x_2 - x_1)$. This may happen with probability $\frac{1}{q}$ and therefore soundness is $\frac{1}{q}$.

Zero-Knowledge: We shall show that Equality of DLOG is honest-verifier zero-knowledge (HVZK). This amounts to showing a simulator that, given the statement to be proven, can produce a transcript whose distribution is indistinguishable from the distribution of the communication between a prover and an honest verifier.
Consider the following simulator, which receives $w = (p, g, h, a, b)$ and a challenge c as input and proceeds as follows:

1. Select $z \in \mathbb{Z}_q$
2. Calculate $u = \frac{g^z}{a^c}$, $v = \frac{h^z}{b^c}$
3. Output (u, v, c, z)

It is easy to see this transcript will always pass the two validation. In addition, in both transcripts (P to V interaction vs. the Simulator's output) c and z are distributed identically and moreover, if $log_g a = log_h b$ the distributions of the entire communication (u, v, c, z) is identical. Otherwise, when $log_g a \neq log_h b$ they are not identical. However, by the DDH assumption $(g, g^\alpha, g^\beta, g^{\alpha\beta})$ is computationally indistinguishable from $(g, g^\alpha, g^\beta, g^\gamma)$ for $\alpha, \beta, \gamma \in \mathbb{Z}_q$. Letting $h = g^\alpha$ this tells us no polynomially bounded verifier V can distinguish between a transcript produced by the simulator, or by a real execution of the protocol. That is, g, h are public, and $u = g^{z-cx_1}$, $v = h^{z-cx_2}$, Consequentially, the DDH assumption tells us that the distribution of (g, h, u, v) where $x_1 = x_2$ is indistinguishable from that where $x_1 \neq x_2$, and so the transcripts are also indistinguishable.

One out of L

Let $g \in Z_q^*$ be some generator and $h \in Z_q^*$ an ElGamal public key. Here, P wishes to prove that one of the l pairs $(x_1, y_1), ..., (x_l, y_l)$ is a re-encryption of the pair (x, y). That is, $\exists t \in \{1..l\}$ such that $(x_t, y_t) = (x, y) \cdot E(1, r) = (xg^r, yh^r)$ (and also $(x_t, y_t) \cdot E(1, -r) = (x, y)$ since re-encryption is symmetric). The protocol is as follows:

1. The prover P selects, for each $i = 1..l$:
   (a) The prover randomly selects $r_i, d_i \in_R Z_q^*$
   (b) Also, let $a_i = (\frac{x_i}{x})^{d_i} g^{r_i}$, $b_i = (\frac{y_i}{y})^{d_i} h^{r_i}$
   (c) The prover computes and records $w = r \cdot d_t + r_t$ for later usage (the prover knows that the t'th pair is a re-encryption)
   (d) **Commitment**: P sends $\{a_1, ..., a_l\}, \{b_1, ..., b_l\}$ to V
2. **Challenge:** V selects $c \in_R Z_q^*$ and sends to P
3. P sends $\{d'_1, ..., d'_l\}, \{r'_1, ..., r'_l\}$ to V where
   (a) $d'_t = c - \sum_{j \neq t} d_j$ and $r'_t = w - r \cdot d'_t$
   (b) for all $j \neq t$, $d'_j = d_j$ and $r'_j = r_j$.

V can now verify the following:

- $c = \sum d'_j$
- for all $i \in \{1..t\}$: $a_i = (\frac{x_i}{x})^{d'_i} g^{r'_i}$ and $b_i = (\frac{y_i}{y})^{d'_i} h^{r'_i}$ (that is, $(a_i, b_i) = ((\frac{x_i}{x})^{d'_i}, (\frac{y_i}{y})^{d'_i}) \cdot E(1, r'_i)$ - an ElGamal re-encryption)

Completeness: The test trivially passes for $i \neq t$. For the t'th pair. Recall that $(x_t, y_t) = (xg^r, yh^r)$, then:

$$(\frac{x_t}{x})^{d'_t} g^{r'_t} = (g^r)^{d'_t} g^{w - r \cdot d'_t} = g^w = (g^r)^{d_t} g^{r_t} = (\frac{x_t}{x})^{d_t} g^{r_t} = a_i$$

and similarly for b_t.

Soundness: Suppose now, none of the pairs are re-encryptions of the others, V can try to guess c in advance to win (with probability $\frac{1}{q}$), otherwise he has to solve two DLOG equations, which might not have a solution.

Zero-Knowledge: TBD.

Non-Interactive ZKP [Fait-Shamir Heuristic]

The two ZKPs above are 3-round protocols and can be transformed into a non-interactive ZKP (NIZKP) using a pseudo-random function h. The proof data and commitment values are hashed, and the result is the challenge $c = h(data, commitment)$. Since h is pseudo-random, there is no need for the verifier to come up with a random challenge. Now P can publish the data, commitment and c so that anyone could verify the veracity of the claim. Practically h is taken as some hash function like $MD5$ or $SHA - 1$, it doesn't really matter as long as it's considered secure and agreed upon before hand.

Example: in one-out-of-L the protocol can be made non-interactive using a challenge $c = h(x, y, x_1, ..., x_l, y_1, .., y_l, a_1, ..., a_l, b_1, ..., b_l)$ along with $d_1, ..., d_l, r_1, ..., r_l$ so anyone can compute $a_1, ..., a_l, b_1, ..., b_l$ to check the validity of c and then continue with the other steps of verification as before.

Voting System: Prêt à Voter

TBD

Appendix: Math Reminder

Lagrange Interpolation

Given $t+1$ data points $(x_0, y_0), ..., (x_t, y_t)$ (where all $x_j's$ are distinct) we would like to obtain a polynomial of degree at most t that passes through all of the data points. The interpolation polynomial in the Lagrange form is a linear combination:

$$L(x) = \sum_{j=1}^{t} y_j l_j(x)$$

Where

$$l_j(x) = \prod_{i \neq j} \frac{x - x_i}{x_{j-} x_i}$$

Note that $l_j(x_j) = 1$ and for all $i \neq j$, $l_j(x_i) = 0$.
There can be only one solution to the interpolation problem since the difference of two such solutions would be a polynomial with degree at most t and $t+1$ zeros. This is only possible if the difference is identically zero, so $L(x)$ is the **unique** polynomial interpolating the given data.
This construction would work just the same in any field $\mathbb{F}_p$ (p a prime number), where division is simply multiplying by the inverse.

Group Theory

Let G be a multiplicative finite group with n elements and e as it's identity element.
Example: The multiplicative group $\mathbb{Z}_n^* = \{a \,|\, a \in \mathbb{Z}^+ \text{ and } gcd(a, n) = 1\}$

Order

The order of G is the number of elements it contains.
We write $ord(G) = |G| = n$.
The order of $a \in G$ is the smallest number $k \in \mathbb{N}$ such that $a^k = e$. In this case, we write $ord(a) = |a| = k$.
Recall that for any such a and k we have:

$$a^k = e \text{ if and only if } ord(a) \text{ divides } k$$

and that $ord(a)|ord(G)$ (according to Lagrange's theorem).

Generator

$g \in G$ is called a generator if $ord(g) = |G|$, or more explicitly if $< g > \triangleq \{g^n \,|\, n \in \mathbb{N}\} = G$. Groups that are generated by a single elements are called

cyclic groups. A group is cyclic iff it has a generator, for example, $\mathbb{Z}_p^*$ is cyclic for all prime p. A concrete example is $\mathbb{Z}_7^*$ where 3 is a generator. Starting from 3 and repeatedly multiplying by 3 (modulo 7) we obtain the set:

$$\{3, 2, 6, 4, 5, 1\} = \mathbb{Z}_7^*$$

We, however, would like to find an element $g \in \mathbb{Z}_7^*$ of order q where q is a Sophie Germain prime. Continuing with our example $7 = 2 \cdot 3 + 1$, here $p = 7$, $q = 3$ and 2 acts as a generator for the subgroup $\{2, 4, 1\} \subset \mathbb{Z}_7^*$ (note that is is of order 3).

Finding a generator of order $q = \frac{p-1}{2}$

Since $\forall a \in \mathbb{Z}_p^*$ $ord(a)|p-1$ (Lagrange), it follows that for $p = 2q + 1$, $ord(a) \in \{1, 2, q, 2q\}$. To find a generator of a subgroup $G \subset \mathbb{Z}_p^*$ with $|G| = q$, we randomly select an element $g \in \mathbb{Z}_p^*$ until $a^2 \neq 1 \bmod p$ and $a^q = 1 \bmod p$.

Modular Inverse

The modular multiplicative inverse of $a \in \mathbb{Z}_n^*$ exists iff a and n are co-prime. The modular inverse of a can be computed using the Extended Euclidean Algorithm:

```
function extended_gcd(a,b)
  if a ≡ 0 mod b
    return {0, 1}
  else {x, y} := extended_gcd(b, a mod b)
  return {y, x-y*(a div b)}
```

The algorithm finds x, y such that $ax + ny = gcd(a, n)$ or rather $ax + ny = 1$ in our case., taking modulo n we see that x is the inverse

Finding Sophie-Germain primes

Given a method M that determines the primality of a number (e.g., Miller-Rabin), it is easy to randomly search for a k-bit Sophie-Germain prime:

1. Randomly select a k-bit long number q (this should take about $O(k) = O(log(q))$ random trials using M)
2. Use M to check if $2q + 1$ is also prime, if not, return to step 1.

Fortunately, it seems there are many Sophie-Germain primes so this random search method usually works quite fast.

Fast exponentiation

Let $a, b \in \mathbb{Z}_p^*$ and consider the task of computing $a^b \bmod p$. A Naive algorithm might work by multiplying $a * a * ... * a$, b times and then take the mod of the result. However this algorithm is problematic in two aspects:

- Space: a^b might require an exponential amount of bits. This problem is easily solved be "pushing" the modulo inside.
- Time: The algorithm has exponential running time (b multiplications, where the input length is $O(log(ab))$.

The *repeated squaring* algorithm overcomes both of these issues.

Chinese Remainder Theorem

Let $n_1, ..., n_k \in \mathbb{N}$ be pairwise co-prime, then for any $a_1, ..., a_k \in \mathbb{Z}$ there exists an $x \in \mathbb{Z}$ solving the following system of equation simultaneously.

$$x \equiv a_i \bmod n_i \ (i = 1, ..., k)$$

Moreover, all solutions are congruent modulo $N = \prod_{i=1}^{k} n_i$.

Proof: To prove the existence of the solution to the system of congruences, we actually describe how to find it:
Suppose there exists integers $e_1, ..., e_k$ such that for all $i, j \in \{1, ..., k\}$ we have:

$$e_j = \begin{cases} 1 \bmod n_i & i = j \\ 0 \bmod n_i & i \neq j \end{cases} \quad (4)$$

Using these integers we could construct:

$$x = \sum_{i=1}^{k} e_i a_i$$

Which is in fact a solution for the system of congruences, as for all $j = 1, ..., k$ we have:

$$a \equiv \sum_{i=1}^{k} e_i a_i \equiv a_j \ (\bmod n_j)$$

since modulo n_j all terms a side of $e_j a_j$ are congruent to 0.
To construct the required k integers $e_1, ..., e_k$, let $N = \prod n_i$ and for all $i = 1, ..., k$ let $N_i = N/n_i$. As all $n_1, ..., n_k$ are pairwise co-prime it follows that $gcd(n_i, N_i) = 1$, so we may use the extended Euclidean algorithm to find integers r_i and s_i such that $r_i n_i + s_i N_i = 1$. Choosing $e_i = s_i N_i$ we notice it satisfies equation (4).

Sophie Germain Primes in light of the Chinese Remainder Theorem

Let p be some prime number. Let $g \in \mathbb{Z}_p^*$ and suppose we'd like to solve the DLOG problem $g^x = y \bmod p$ for some given constant $y \in \mathbb{Z}_p^*$.
Using the brute-force approach will take up $O(p)$ time, which is exponential in the length of the input $O(log(p)$.
Suppose now, that we know the factorization of $p-1 = \prod_{i=1}^{k} q_i$. A (possibly) more effective way to find x (the Pohlig-Hellman algorithm) is to separate the single equation to k equations, each of the form:

$$(g^x)^{\frac{p-1}{q_i}} = y^{\frac{p-1}{q_i}} \bmod p \ (i = 1, ..., k)$$

Now, brute-forcing each of these equations requires $O(q_i)$ steps, and in total $O(\sum_{i=1}^{k} q_i)$ which is significantly less than $O(p)$. In fact we have found the value of x modulo each of the $q_i's$, and so we may now use the Chinese Remainder Theorem to find x itself.

Finally, notice that when $p = 2q + 1$ (q is prime) this algorithm has no real advantage over the naive brute-force. Both will take $O(p)$ number of steps.

Carmichael's Function

Let n be a positive integer, the Carmichael function $\lambda(n)$ is defined as a smallest positive m such that:

$$a^m = 1 \bmod n$$

for every integer a that is co-prime to n.
That is $a^{\lambda(n)} \equiv_n 1$ for all a such that $gcd(a, n) = 1$.

This function can be defined as follows:

$$\lambda(n) = \begin{cases} 1 & \text{for } n = 2 \\ 2 & \text{for } n = 4 \\ 2^{k-2} & \text{for } n = 2^k \text{ where } k \geq 3 \\ p^{k-1}(p-1) & \text{for } n = p^k \text{ where } p \geq 3 \text{ is prime} \\ lcm(\lambda(p_1^{k_1}), ..., \lambda(p_t^{k_t}) & \text{for } n = \prod p_i^{k_i} \text{ where } p_i \text{ are distinct primes} \end{cases}$$

References

Papers and Class Notes

- http://www.cs.tau.ac.il/~amnon/Students/ben.riva.thesis.pdf
- Ronald Cramer, Rosario Gennaro, Berry Schoenmakers, A Secure and Optimally efficient Multi-Authority Election Scheme
- **Paillier**: Pascal Paillier, Public-Key Cryptosystem Based on Composite Degree Residuosity Classes
- **Equality of DLOG**: David Chaum and Torben P. Pedersen. Wallet databases with observers. In CRYPTO, pages 89-105, 1992.
- **ZKP**: http://cs.nyu.edu/courses/spring07/G22.3220-001/lec3.pdf
- **Carmichael Function**: http://reflections.awesomemath.org/2007_2/carmichael.pdf
- **Inverting DLOG**: http://www.nada.kth.se/kurser/kth/2D1449/krypto06/PDF/lecture7.pdf

Wiki

- http://en.wikipedia.org/wiki/Discrete_logarithm
- http://en.wikipedia.org/wiki/ElGamal
- http://en.wikipedia.org/wiki/Zero-knowledge_proof
- http://en.wikipedia.org/wiki/Exponentiation_by_squaring
- http://en.wikipedia.org/wiki/Chinese_remainder_theorem
- http://en.wikipedia.org/wiki/Carmichael_function