PARSEKEY:
GÜVENLİ KİMLİK BELİRLEME HİZMETİNDE YENİ BİR YAKLAŞIM

Behnam RAHNAMA; Atilla ELÇİ
Bilgisayar Mühendisliği Bölümü ve İnternet Teknolojileri Araştırma Merkezi, Doğu Akdeniz Üniversitesi,
Gazimagusa, Mersin 10, Turkey
{behnam.rahnama; atilla.elci }@emu.edu.tr

ÖZET
Güvenli olmayan ağlarda kimlik belirleme hizmetini kırmaya yönelik şifre çözümlemesi korunmuş bilgilerin elde edilmesi olasılığına işaret ediyor. Buna göre araştırmacılar daha güvenli temelleri geliştirip uygulamaya çalışmaktadırlar. Gizli yazım (*steganography*) yoluyla MD5 ve SHA1 algoritmaları uygulanarak anahtarın gizlenmesi temelinde yeni ve güvenli bir yöntem olarak ParseKey önerilmektedir. Bu yöntem aynı zamanda, istemci tarafında giriş bilgilerini tutmak veya sunumcu VT'ndaki anahtarı değiştirme gereği gibi zayıflıkları öngörmüyor.

ABSTRACT
Cipher analysis on the authentication services through insecure network shows possibility of illegal capture of protected information, researchers try to develop and implement safer foundations. ParseKey represents a new approach to highly secure and safe authentication method by applying steganography to hide MD5 and SHA1 algorithms- encrypted keys, which are quite secure as well and furthermore having possibility of uniqueness on signed in users or agents by changing the ParseKey for each of them at login time without suffering of holding the users' side login data or changing the key in server side DB.

KEY WORDS
ParseKey, cryptography and authentication, steganography and encryption algorithm.

ANAHTAR SÖZCÜKLER
ParseKey, kriptografi ve kimlik belirleme, gizli yazım (*steganography*) ve şifreleme algoritması.

1. Giriş

Güvenli korunmuş bilgiye erişmede, güvensiz ağ ve platformlardaki istemleri denetlemek ve yönetmek için kimlik belirleme hizmetine (*authentication service*) [10] gerek vardır. Hizmet, kullanıcı adı ve parolası gibi giriş bilgisini ve belki başka durağan veriyi denetler ve sunumcudaki verilerle uyuşması durumunda yetki verir. Bu noktada, gerek durağan veri tutuluyor olması gerekse şifreleme algoritmasının bilinmesi sonucunda, şifre çözümlemesi yoluyla sisteme saldırılar olur. Keza, gizli ikincil girişleri denetlemek hiç de kolay değildir; kimi zamanlarda bilgi sistemi altyapısı ve temel nedenlerden dolayı mümkün de olmayabilir.

Bu bildiride önce ParseKey şifreleme ve çözme (*encryption and decryption)* algoritmasının çalışması gözden geçirilecek, sonraki bölümde uygulanması kısaca gösterilecektir. Çözümleme yoluyla algoritmaya nasıl saldırılabileceği ve olası zayıflıklarının nasıl giderileceği ele alınacaktır. Böylece, ParseKey adıyla anılan kırılmaz güvenlikli kimlik belirleme hizmeti oluşturulacaktır. Sonuçta, pratik uygulama örneklerine yer verilecektir.

2. ParseKey algoritması nasıl çalışır?

ParseKey şifreleme ve çözme algoritmasının açıklaması için önce veritabanı gereği ve anahtar türetme üzerine birkaç tanım gereklidir.

2.1. Tanımlar

Parola ve ParseKey gereksinimi veriler sunumcudaki veritabanında (VT) saklanır. Kullanıcı hizmete eriştiğinde giriş penceresi görünür; ad ve parola girildiğinde ParseKey verisi sunumcuya gönderilir. Kimlik belirleme hizmeti bu noktada istemciden yüklenen veriyi kendinde saklı veri ile karşılaştırır. Tam uyuşma durumunda girişe izin verilir ve yeni ParseKey verisi istemciye indirlir. Gelecek kez sisteme girişte bu veri kullanılacaktır.

2.2. Şifreleme ve Çözme Yöntemi

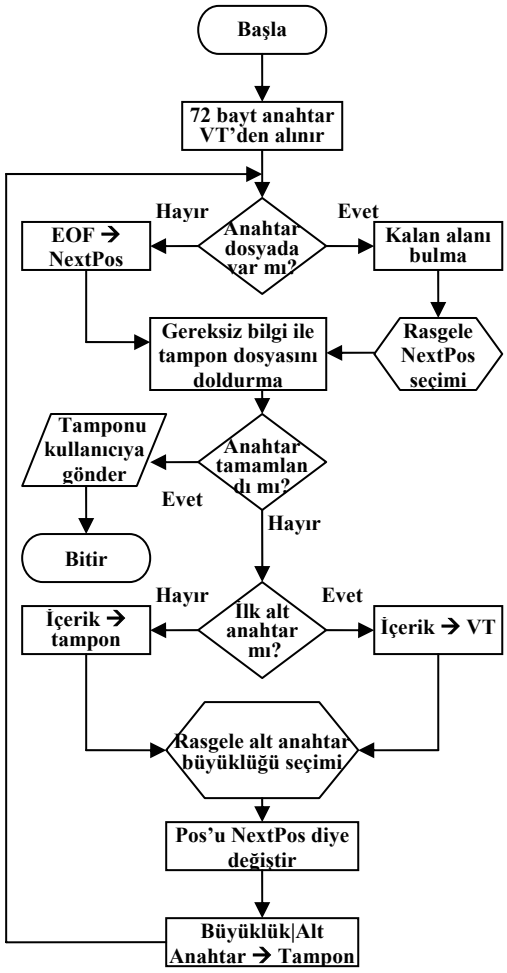
Sisteme girişlerde kullanılan adlar veya kimlik numaraları birik olduğundan, bunlardan birik anahtarlar üretilebilir. Bu amaçla ParseKey yönteminde MD5 [1] ve SHA1 [1] algoritmaları ayrı ayrı kullanılır. Doğallıkla MD5 işlevi 32 baytlık ve SHA1 işlevi ise 40 baytlık onaltılı dizisi döndürür. Bunlar bitiştirilince 72 baytlık onaltılı dizisinde oldukça güvenli bir anahtar elde edilir[1].

ParseKey yönteminde MD5 ve SHA1 kullanılarak elde edilen anahtarlar, algoritmanın yarattığı gürültü veri içine düzenli dağıtılmıştır. Öyle ki, pratik olmamakla birlikte, kuramsal olarak MD5 ve SHA1 algoritmaları kesinlikle çözülemez değilseler [6] bile, düzenli dağılım verdikleri için ParseKey yönteminde yararlıdırlar. Olası bir saldırgan ParseKey kütüğü içinde önce anahtarı belirlemelidir ve ancak ondan sonra MD5 ve SHA1 şifre çözümüne girişebilir.

Sonraki konu anahtar içinde bu iki algoritmanın anahtarlarının sırasıdır. Bu algoritmalar ardışık kullanım, yani birinin çıktısının diğerinin girdisi olarak kullanılması, sonuçta daha kısa anahtar verecektir. Ancak, karmaşıklık aynı kalacaktır. Sıra önemli değildir, her iki şifreleme algoritmasının karmaşıklığı eklenmektedir [1, 2].

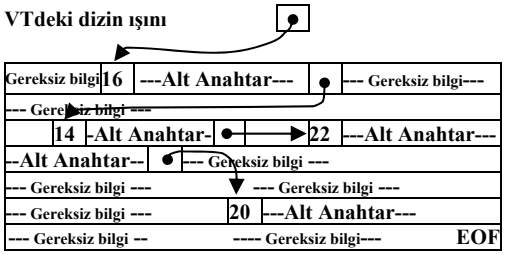
ParseKey'nin VT'nda tutulan değiştirgen (*parameter*) değerinden biri yaratılıp istemciye iletilen ParseKey anahtar verisi ve diğeri ise anahtar kıyım dizinleri

ışınıdır. ParseKey verisi girişte anılan kütüktür; anahtar kıyım dizinlerinden ise ileride söz edilecektir.
ParseKey istemci kütüğünde, parçalanan anahtar alt anahtarlar olarak rasgele dağıtılmıştır. Bunların arasına ise düzgün rasgele dağıtılmış onaltılı çöp (gereksiz bilgi, gürültü, [3]) baytları yerleştirilmiştir. Kimlik belirleme hizmeti dizin ışınını kullanarak anahtarı elde eder ve VT'deki ile karşılaştırır [4].
Alt anahtar dağıtım yöntemi, biri alt anahtar boyunu belirleyen diğeri ise alt anahtarın ParseKey kütüğündeki yerini belirleyen iki rasgele sayı türetir. İlk akla gelen gerçekleme modeli, ilk konum bilgisini dizin ışınında tutuktan sonra diğerlerini ParseKey kütüğünde tutmaktır. Burada her alt anahtar paketi üç farklı veri içerir: paket boyu, veri, ve sonraki pakete bağlantı. Şekil 2.1 bu şifreleme yöntemini gösteriyor.



Şekil 2.1: ParseKey şifreleme akışı: ParseKey kütüğünün oluşturulması.

Anahtar çözme yöntemi, sunucu veri kaydındaki dizin değerini kullanarak istemciden yüklenen ParseKey verisinin içinden ilk alt anahtar paketini çeker. Yukarıda belirtildiği gibi, paketin ilk bileşeni alt anahtar boyudur. Sonraki bileşeni alt anahtar verisi ve sonuncusu da sıradaki alt anahtar paketinin yerini gösteren bir referanstır. Böylece toplam 72 bayta kadar olan diğer alt anahtarlar da elde edilir. Örnek bir ParseKey kütüğü Şekil 2.2'de gösterilmiştir.



Şekil 2.2: ParseKey kütüğü genel yapısı- ilk alt anahtar konum değeri VTde diğerleri kütükte tutulur.

2.3. ParseKey uygulaması

Ekte verilen PHP programı ParseKey kimlik belirleme şifreleme ve çözme algoritmasının gerçekleştirimidir. İşlevler ayrı ayrı ele alınmıştır.
ParseKey algoritmasının programlanmasında hazır kütüphanelerden yararlanmak mümkündür. MD5 ve SHA1 için gerekli sınıf kütüphaneleri .NET Framework [7] ve PHP [8] içinde bulunmaktadır. Yapılacak iş sadece ilgili işlevleri uygulama içinden uygun değiştirgenlerle çağırmaktır.

2.4. Sunulan algoritmanın şifre çözümlemesi [5]

Yukarıda anlatılan ilk kuşak ParseKey'nin veri paketlerine bağlantısının yalın bir bağ olduğunu bilen rasgele korsanlar, ilk alt anahtar dizinini tahmin edip anahtarı yakalamayı deneyebilirler. Örneğin, ParseKey'nin bir kilo baytlık kütük boyu, 1024(kütük boyu) - 72(parçalanmamış anahtar boyu) - 2(anahtar boyu gösterim uzunluğu) anahtarı keşfetmenin 950 değişik seçeneğini verir. Bu seçeneklerden birinde ilk iki bayt alt anahtar paket boyunu, paketin son dört baytı sonraki alt anahtar paketinin dizinini ve tabii arada kalan da alt anahtarın kendisini verir. Böylece birkaç öteleme ile 72 bayt anahtarın tümü keşfedilebilir. Ki bu 950 seçeneğin bir çoğunda ya anahtar boyu 72 çıkmayabilir ya da alt anahtar bağı kütük dışı veya geriye atıf olabilir. Bu seçenekler hemen atılır. Geriye kalan az sayıda seçenek ise kolaylıkla çözülür.
Uygulanan saldırı işlevi aşağıda verilmiştir.

```
function find_key(){
 $filename = "ParseKey.dat";
 clearstatcache();
 $handle = fopen ($filename, "r");
 fseek($handle, 0);
 $contents = fgets($handle, filesize
        ($filename)+1);
 fclose ($handle);
 echo "Start Decoding key ...<br>";
 $firstindex=0;
 $newkey='';
 while ($firstindex<952){
  $newkey = decode_key ($firstindex,
    $contents);
  if    (strlen($newkey)==72)    echo
    "<b>key          of          index
```

```
    ".$firstindex."is:".
    $newkey."<br></b>";
  $firstindex++;
 }
}
```

Şifreleme, saldırı ve çözme işleminin sonuçları aşağıdaki çıktıda verilmiştir. Aday anahtarlardan biri ParseKey kütüğüne saklanmış gerçek anahtardır.

Key:
D8932FD49BB05FF0670C3545D15547CB7BAA
F2087F7D49795DCF0A82605FB1103ED20D28

Success, 1024 bytes wrote to file (ParseKey.dat)

Start Decoding key...

Key of index 314 is:
D8932FD49BB05FF0670C3545D15547CB7BAA
F2087F7D49795DCF0A82605FB1103ED20D28

Start Reading...

File size: 1024 bytes

Contents:
6BA352AC19894162E9203D08D899B2735317
F0F21676B6077F91138ACB83039D6489B32F
B350F118F01C8C77F4B9DB3FA89955841B63
FFB49D21178BCE129A14340F8BDB4E580F25
3B82FDA14D799878372119E7D7AAB11C9FF8
D394A1289BF24801CCBBE1B145157B90CA96
FE52CB9F109CC5095ADB35756B2B73CEB1A3
FF2811B2CFD01E2BD38375668D1CD8932FD4
9BB05FF0670C3545D15501F019F34E5EC09A
F81016099CFB09DD83133A880972575C…

Start Decoding key...

Key of index 101 is:
9B32FB350F118F01C8C77F4B9DB3FA899558
41B63FFB49D21178BCE129A14340F8BDB4E5

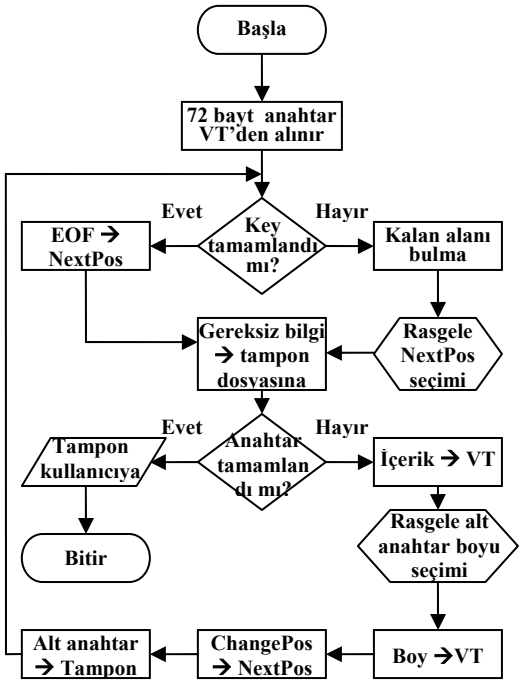
Key of index 228 is:
01CCBBE1B145157B90CA96FE52CB9F109CC5
095ADB35756B2B73CEB1A3FF2811B2CFD01E

Key of index 314 is:
D8932FD49BB05FF0670C3545D15547CB7BAA
F2087F7D49795DCF0A82605FB1103ED20D28

Key of index 729 is:
35E3CFC90C6F590981EF1B51C684E606A653
4B149800AA24E1A5E9DDC81F1A31573381EB

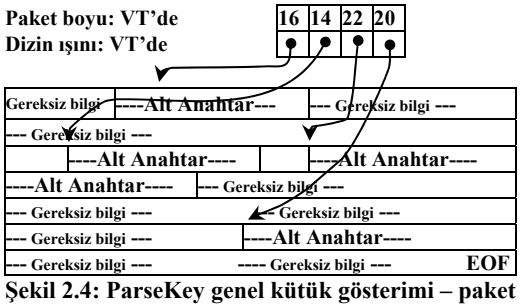
2.5. Kırılmaz güvenli kimlik belirleme algortimasının elde edilişi

Bu sorunu gidermek için algoritmayı iyileştirmek gerekiyor. Bu gelişmeleri yapmak kolaydır. birinci ve en önemlisi tüm tüm alt anahtar dizinlerini ve boylarını VTde dizin ışını olarak tutmaktır. Böylece kütük içindeki bağ gözardı edilecek, ve bunun sonucunda şifre saldırı yöntemleri daha karmaşıklaşacaktır. İkinci yapılacak ise alt anahtar paketlerini ParseKey kütüğü içinde birbirinden bağımsız olarak rasgele dağıtmaktır. Bu geliştirmelerin öncesi ve sonrasında elde edilen algoritmaları karşılaştırınca, zayıf ve güvensiz bir algoritmadan güvenli ve kırılmaz bir uygulama elde edildiği görülmektedir. Geliştirilmiş nitelikli ParseKey algoritmasının iş akış biçim Şekil 2.3'de verilmiştir.



Şekil 2.3: Kırılmaz ParseKey algoritmasının iş akış çizimi.

Şekil 2.4 ise ParseKey kütüğünün yeni gösterimi verilmiştir. Burada bağların ve paket boylarının VT'de saklanması şık bir çözüm oluşturuyor.



Şekil 2.4: ParseKey genel kütük gösterimi – paket boyu ve konum bağı VT'dedir; alt anahtarlar kütükte dağıtık olarak rasgele seçilmiş gereksiz veri arasına serpiştirilmiştir.

3. ParseKey algoritmasının verimliliği üzerine inceleme sonuçları

ParseKey algoritması, yer değiştirme (*transposition*) ve gizli yazım [9] yoluyla veri akışını şaşırtmaya dayanır. ParseKey kütüğünün kimi özellikleri bulunması gerekir. Kütük boyu, alt anahtar paketlerinin anahtar içindeki yerini değiştirmek için, en az 72 bayt olabilir. Bu yeterli güvenli olmayacaktır, çünkü sadece $16^{72} = 4.973232641\ E^{86}$ ayrı çözüm vardır, bu sonuç ise 168 ikilli güvenlik anahtarına saldıran şifre kırıcıları için bile küçümsenecek bir değer değildir. Yine de, daha büyük kütük boyu çok daha karmaşıklık sağlayacaktır. Örneğin, bir kilobayt kütük boyu ile $16^{1024} = 1.044388881\ E^{1233}$ ayrı çözüm sunar ki "oldukça" kırılamaz bir algoritmadır. Veri akışının gerektirdiği dizin ışını VT'de saklanır ve alt anahtar boyu ve bağı için en az $72 \times 2 + 72 \times 4 = 432$ bayttır. Böylece kullandığı alan olarak algoritmanın maliyet verimliliği $72/(72+432) \approx 14.2\%$'dir [4]. Böyle düşük bir verimlilik oranı veri aktarımı için uygun olmamakla birlikte, DES gibi diğer algoritmaların güvenlik anahtarı gibi çok önemli verinin aktarılmasında oldukça iyidir. Ayrıca, daha uygun bir kullanım alanı ise açık ağlarda kimlik belirleme ve doğrulama hizmetleridir.

Anma

Bu çalışmadaki destekleri için Doğu Akdeniz Üniversitesi, Internet Teknolojileri Araştırma Merkezi'nde Alexander Chefranov ile ETec Araştırma Enstitüsü'nde Mehdi Veissi, Mahyar Mostowfi, ve Karim Khalifeh'ye teşekkür ederiz.

Kaynaklar

[1] W. Stallings, *Cryptography and Network Security Principles ad practices 3rd Ed.*, Pearson Education: Prentice Hall, 2001.

[2] A. Eskicioglu, L. Litwin, Cryptography, *International Journal of Potentials, IEEE, 20*(1), 2001, 36-38.

[3] D. Artz, Digital steganography: hiding data within data, *International Journal of Internet Computing, IEEE, 5*(3), 2001, 75-80.

[4] M.M. Amin, M. Salleh, S. Ibrahim, M.R. Katmin, M.Z.I. Shamsuddin, Information hiding using steganography, *Proc. 4th National Conf. on Telecommunication Technology*, NCTT 2003, 21-25.

[5] Li Zh, Sui Ai Fen, Yang Yi Xian, A LSB steganography detection algorithm, *Proc. 14th IEEE Conf. on Personal, Indoor and Mobile Radio Communications*, PIMRC 2003, 2780-2783 vol.3.

[6] ACM TechNews, *Volume 7, Issue 766 Mart, 2005* http://www.acm.org/technews/articles/20057/0316w.html#item5. 7 Mart 2005 günü ziyaret edildi.

[7] Microsoft Developer Network, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemsecuritycryptographymd5classtopic.asp. 7 Mart 2005 günü ziyaret edildi.

[8] PHP Documentations, http://tr2.php.net/manual/en/function.md5.php. 7 Mart 2005 günü ziyaret edildi.

[9] Gizli yazım, *Wikipedia*, the free encyclopedia, http://en.wikipedia.org/wiki/Steganography; and in Google, http://www.google.com.tr/search?hl=tr&q=define%3Asteganography&btnG=Ara&meta=. 7 Mart 2005 günü ziyaret edildi.

[10] G. V Van Rooij, *Authentication services: Theory and practice*, Technische Hogeschool Eindhoven, 1989. ASIN: B0007CBTP6

Appendix

```
//Initial definitions such as ParseKey file size and Key size.
define(MAXBAND,1000000);
define(MAXLINKSIZE,500);//KEYSIZE*6+KEYSIZE-4
define(MAXFILESIZE,1024);
define(KEYSIZE,72);

//this function makes uniformly random hexadecimal character stream as
garbage data using in gaps between sub keys.
function make_garbage($len){
  $strout='';
  while(strlen($strout)<$len) {
    $selector=rand(0,1);
    if($selector==1)
      $strout.=md5(rand(1,MAXBAND));
    else
      $strout.=sha1(rand(1,MAXBAND));
  }
  return substr($strout,0,$len);
}

//sub key packets are made with make_paket function.
function make_packet($key,$start,$len,$nextindex){
  if($start+$len>KEYSIZE) {echo "Range Error "; exit; }
  $packet=dechex($len);
  if (strlen($packet)==1) $packet="0".$packet;
  $packet.=substr($key,$start,$len);
  $hexindex=dechex($nextindex);
  if ($nextindex!=0)
    while(strlen($hexindex)<4)
      $hexindex="0".$hexindex;
  else
    $hexindex='';
  $packet.=$hexindex;
  return $packet;
}

//to has a unique and hexadecimal highly secure key MD5 and SHA1 algorithms
are applied.
function make_key($username){
  return sha1($username).md5($username);
}

//the ParseKey file may contains some header information as well.
function make_header(){
  return "ParseKey, a new approach to unbreakable secure authentication
service";
}

//encryption algorithm engine is compacted in the encode_key role.
function encode_key($username){
  $key=make_key($username);
  echo "<b>key : ".$key."<br></b>";
  $buffer=make_header();
  $oldlinksize=0;
  $sumkeyparts=0;
  $minposrange=strlen($buffer);
  $maxposrange=MAXFILESIZE-MAXLINKSIZE;
  $index=rand($minposrange,$maxposrange);
  //write data entity into DB index field
```

```
  global $firstindex;
  $firstindex=$index;
  while($sumkeyparts<KEYSIZE){
    $buffer.=make_garbage($index-strlen($buffer));
    $minkeyrange=1;
    $maxkeyrange=KEYSIZE-$sumkeyparts;
    $partsize=rand($minkeyrange,$maxkeyrange);
    $oldlinksize+=(6+$partsize);
    $minposrange=strlen($buffer)+(6+$partsize);
    $maxremain=7*(KEYSIZE-($sumkeyparts+$partsize))-4;
    $maxposrange=MAXFILESIZE-$maxremain;
    if($sumkeyparts+$partsize==KEYSIZE)
      $index=0;
    else
      $index=rand ($minposrange,$maxposrange);
    $buffer.=make_packet($key,$sumkeyparts,$partsize,$index);
    $sumkeyparts+=$partsize;
  }
  $buffer.=make_garbage(MAXFILESIZE-strlen($buffer));
  return $buffer;
}

//accessing the key which is hided in ParseKey file by following the link
and applying the first index, is quite possible.
function decode_key($index,$buffer){
  $key='';
  while(strlen($key)<KEYSIZE){
    $len=hexdec(substr($buffer,$index,2));
    $key.=substr($buffer,$index+2,$len);
    $index=hexdec(substr($buffer,$index+2+$len,4));
  }
  return $key;
}

//an example on encryption and decryption functions.
$filename = "ParseKey.dat";
if (!$handle = fopen($filename, 'w')){
  print "Cannot open file ($filename)";
  exit;
}
$len=fwrite($handle,encode_key($username));
if (!$len){
  print "Cannot write to file ($filename)";
  exit;
}
fclose($handle);
clearstatcache();
$handle = fopen ($filename, "r");
fseek($handle, 0);
$contents = fgets($handle, filesize($filename)+1);
fclose ($handle);
echo "contents : $contents</b>";
echo "Start Decoding key ...<br>";
echo "<b>key : ".decode_key($firstindex,$contents)."</b>";
```