Seguridad y Protección de Sistemas Informáticos

Fco. Javier Lobillo Borrero

Departamento de Álgebra, Universidad de Granada

Curso 2017/2018

Índice

Índice

Sustitución monoalfabética

Denotamos por $\mathcal{A}$ un alfabeto, y $\mathcal{A}^*$ el conjunto de las cadenas sobre el alfabeto de longitud arbitraria. Una sustitución monoalfabética es una aplicación biyectiva

$$e : \mathcal{A} \to \mathcal{A},$$

que se extiende a las cadenas de forma natural

$$e : \mathcal{A}^* \to \mathcal{A}^*, [e(x_0x_1\cdots) = e(x_0)e(x_1)\cdots]$$

Cifrado de César

$\mathcal{A} = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{G}, \mathtt{H}, \mathtt{I}, \mathtt{K}, \mathtt{L}, \mathtt{M}, \mathtt{N}, \mathtt{O}, \mathtt{P}, \mathtt{Q}, \mathtt{R}, \mathtt{S}, \mathtt{T}, \mathtt{V}, \mathtt{X}, \mathtt{Y}, \mathtt{Z}\}$, $e : \mathcal{A} \to \mathcal{A}$ un desplazamiento cíclico a la derecha de tres posiciones. Matemáticamente

$$e : \mathbb{Z}_{23} \to \mathbb{Z}_{23}, [e(x) = x + 3 \pmod{23}].$$

Sustitución monoalfabética: Ejemplos I

Criptosistema de desplazamiento

```
sage: A = AlphabeticStrings()
sage: S = ShiftCryptosystem(A)
sage: K = 7
sage: P = A.encoding("El criptosistema de desplazamiento generaliza al cifrado de César"); P
ELCRIPTOSISTEMADEDESPLAZAMIENTOGENERALIZAALCIFRADODECSAR
sage: C = S.enciphering(K,P); C
LSJYPWAVZPZALTHKLKLZWSHGHTPLUAVNLULYHSPGHHSJPMYHKVKLJZHY
sage: S.deciphering(K,C)
ELCRIPTOSISTEMADEDESPLAZAMIENTOGENERALIZAALCIFRADODECSAR
sage: pdict = S.brute_force(C); pdict
{0: LSJYPWAVZPZALTHKLKLZWSHGHTPLUAVNLULYHSPGHHSJPMYHKVKLJZHY,
1: KRIXOVZUYOYZKSGJKJKYVRGFGSOKTZUMKTKXGROFGGRIOLXGJUJKIYGX,
2: JQHWNUYTXNXYJRFIJIJXUQFEFRNJSYTLJSJWFQNEFFQHNKWFITIJHXFW,
...
6: FMDSJQUPTJTUFNBEFEFTQMBABNJFOUPHFOFSBMJABBMDJGSBEPEFDTBS,
7: ELCRIPTOSISTEMADEDESPLAZAMIENTOGENERALIZAALCIFRADODECSAR,
8: DKBQHOSNRHRSDLZCDCDROKZYZLHDMSNFDMDQZKHYZZKBHEQZCNCDBRZQ,
... }
```

Sustitución monoalfabética: Ejemplos II

Sustitución monoalfabética

```
sage: E = SubstitutionCryptosystem(A)
sage: P = E.encoding('Este es un ejemplo de una sustitucion monoalfabetica general. Como pretendemos realizar un
primer ejemplo de ataque encaminado a la ruptura vamos a utilizar un ejemplo de cierto tamano.')
sage: K = A('JUCFLPKQWGVTXYBRMDZIEOAHSN'); K
JUCFLPKQWGVTXYBRMDZIEOAHSN
sage: e = E(K)
sage: C = e(P); C
LZILLZEYLGLXRTBFLEYJZEZIWIECWBYXBYBJTPJULIWCJKLYLDJTCBXBRDLILYFLXBZDLJTW
NJDEYRDWXLDLGLXRTBFLJIJMELLYCJXWYJFBJTJDERIEDJOJXBZJEIWTWNJDEYLGLXRTBFLC
WLDIBIJXJYB
sage: E.deciphering(K,C)
ESTEESUNEJEMPLODEUNASUSTITUCIONMONOALFABETICAGENERALCOMOPRETENDEMOSREALI
ZARUNPRIMEREJEMPLODEATAQUEENCAMINADOALARUPTURAVAMOSAUTILIZARUNEJEMPLODEC
IERTOTAMANO
```

Sustitución monoalfabética: Ejemplos III

```
sage: C.frequency_distribution()
Discrete probability space defined by {P: 0.00645161290322581, W: 0.0580645161290323,
B: 0.0838709677419355, X: 0.0645161290322581, C: 0.0322580645161290, I: 0.0645161290322581,
D: 0.0645161290322581, J: 0.122580645161290, E: 0.0645161290322581, K: 0.00645161290322581,
F: 0.0322580645161290, L: 0.148387096774194, G: 0.0193548387096774, R: 0.0387096774193548,
M: 0.00645161290322581, N: 0.0129032258064516, Y: 0.0709677419354839, T: 0.0516129032258065,
O: 0.00645161290322581, Z: 0.0387096774193548, U: 0.00645161290322581}
```

Sustitución polialfabética

Dadas t biyecciones en el alfabeto $\mathcal{A}$

$$e_0, \ldots, e_{t-1} : \mathcal{A} \to \mathcal{A},$$

se define una sustitución polialfabética como la aplicación biyectiva

$$e : \mathcal{A}^* \to \mathcal{A}^*, [e(x_0 \ldots x_{t-1} x_t \ldots x_{2t-1} x_{2t} \ldots) = e_0(x_0) \ldots e_{t-1}(x_{t-1}) e_0(x_t) \ldots e_{t-1}(x_{2t-1}) e_0(x_{2t}) \ldots].$$

Cifrado de Vigenere

Las biyecciones $e_i : \mathcal{A} \to \mathcal{A}$ son cifrados de desplazamiento, determinados por las posiciones, empezando por 0, que ocupan las letras de una palabra clave. Por ejemplo, una palabra clave ABCBA se corresponde con las funciones

$$\begin{aligned} e_0(x) &= x \pmod{26}, \\ e_1(x) &= e_3(x) = x + 1 \pmod{26}, \\ e_2(x) &= x + 2 \pmod{26}, \\ e_3(x) &= x + 1 \pmod{26} \\ e_4(x) &= x \pmod{26}. \end{aligned}$$

Cifrado de Vigenere: Ejemplo I

```
sage: A = AlphabeticStrings()
sage: E = VigenereCryptosystem(A,6)
sage: K = A('JAVIER')
sage: P = A.encoding("Rugby union, or simply rugby, is a contact team sport which originated in England in the first
half of the 19th century.[3] One of the two codes of rugby football, it is based on running with the ball in hand. In
its most common form, a game is between two teams of 15 players (two more than rugby league) using an oval-shaped
ball on a rectangular field with H-shaped goalposts on each try line. In 1845, the first football laws were written
by Rugby School pupils; other significant events in the early development of rugby include the Blackheath Club's
decision to leave the Football Association in 1863 and the split between rugby union and rugby league in 1895.
Historically an amateur sport, in 1995 restrictions on payments to players were removed, making the game openly
professional at the highest level for the first time.[4]")
sage: C = E.enciphering(K,P); C
AUBJCLWIJVSIBIHXPPAUBJCZBAXWRKJCOBIRVSKWVKFHDKLFAIBQRRCEYQRVWGGIRURNOPIWRRNBLRUFJNXYNTCK
IECUMGSENOABLVCWJKSUNSJNVLPBTNSFCBVTPZCINJEJNDJVVLWNDVKNRTCBLVKAGTMEQAILMERTNUSJCCJUQFWF
JZQRPAHMMJKEOEIVWTRWXVJMNWJGUATMVJCWJUSINTCIRIDGWGPVJGPMYJRNBIRFEAGALRYEYJECUOIIVVLTVVKL
UAMNMVUDRQXYQSCITVMGJIPGXSOASENAXPXIHLDVIZWTCMJZASONSFCBVTPCJWNEIINWMQXKNNWGVLPBTAGYXOGX
YGRLNWXYNRNQKERFDKEECEQMRKBIIBLVNAMTCUNVZTSGVEIBSWAUBJCZWCGCHVCHZJPRLKCMEKQCGCFJMEXQWZXN
OWPVJVZBLVOOJBFRULVAWFLIVBMFWIIIRUCHZATCRTWMXNNEIZYXKYPVMFWAILVLPBTTIRPUZQRYRSOWVZLAGTCR
WAHIXVDRNXSICIIZIJCRDKXZXNNWRGJYHMRKBTJXPRHEMAAVAEMMQFEEYUEBRNBBLVPAHMSGNNGGTIXFZAWZXNVT
EKCHZPMXQENBPVEEGNSICHZNMIBTOQQV
sage: E.deciphering(K,C)
RUGBYUNIONORSIMPLYRUGBYISACONTACTTEAMSPORTWHICHORIGINATEDINENGLANDINTHEFIRSTHALFOFTHETHC
ENTURYONEOFTHETWOCODESOFRUGBYFOOTBALLITISBASEDONRUNNINGWITHTHEBALLINHANDINITSMOSTCOMMONF
ORMAGAMEISBETWEENTWOTEAMSOFPLAYERSTWOMORETHANRUGBYLEAGUEUSINGANOVALSHAPEDBALLONARECTANGU
LARFIELDWITHHSHAPEDGOALPOSTSONEACHTRYLINEINTHEFIRSTFOOTBALLLAWSWEREWRITTENBYRUGBYSCHOOLP
```

Cifrado de Vigenere: Ejemplo II

```
UPILSOTHERSIGNIFICANTEVENTSINTHEEARLYDEVELOPMENTOFRUGBYINCLUDETHEBLACKHEATHCLUBSDECISION
TOLEAVETHEFOOTBALLASSOCIATIONINANDTHESPLITBETWEENRUGBYUNIONANDRUGBYLEAGUEINHISTORICALLYA
NAMATEURSPORTINRESTRICTIONSONPAYMENTSTOPLAYERSWEREREMOVEDMAKINGTHEGAMEOPENLYPROFESSIONAL
ATTHEHIGHESTLEVELFORTHEFIRSTTIME
sage: C.frequency_distribution()
Discrete probability space defined by V: 0.0648148148148148, X: 0.0370370370370370,
J: 0.0493827160493827, L: 0.0354938271604938, N: 0.0632716049382716, Y: 0.0216049382716049,
P: 0.0324074074074074, B: 0.0432098765432099, D: 0.0138888888888889, F: 0.0246913580246914,
Q: 0.0262345679012346, H: 0.0200617283950617, S: 0.0293209876543210, U: 0.0293209876543210,
W: 0.0462962962962963, I: 0.0632716049382716, K: 0.0308641975308642, M: 0.0416666666666667,
O: 0.0200617283950617, Z: 0.0308641975308642, A: 0.0432098765432099, C: 0.0540123456790123,
E: 0.0462962962962963, G: 0.0370370370370370, R: 0.0570987654320988, T: 0.0385802469135802
sage: aux = A('')
sage: for ii in range(len(C)):
... if ii% 4 == 0:
... aux *= C[ii]
...
sage: aux.frequency_distribution()
Discrete probability space defined by V: 0.0740740740740741, X: 0.0493827160493827,
J: 0.0617283950617284, L: 0.0432098765432099, N: 0.0864197530864197, Y: 0.0308641975308642,
P: 0.0370370370370370, B: 0.0370370370370370, D: 0.0246913580246914, Q: 0.0308641975308642,
H: 0.0246913580246914, S: 0.0185185185185185, U: 0.0246913580246914, W: 0.0493827160493827,
I: 0.0493827160493827, K: 0.0123456790123457, M: 0.0555555555555556, O: 0.0308641975308642,
Z: 0.0370370370370370, A: 0.0246913580246914, C: 0.0493827160493827, E: 0.0185185185185185,
```

Cifrado de Vigenere: Ejemplo III

```
G: 0.0246913580246914, R: 0.0802469135802469, T: 0.0246913580246914
sage: aux = A('')
sage: for ii in range(len(C)):
... if ii% 6 == 0:
... aux *= C[ii]
...
sage: aux.frequency_distribution()
Discrete probability space defined by P: 0.0555555555555556, V: 0.0185185185185185,
A: 0.0555555555555556, W: 0.0925925925925926, B: 0.0462962962962963, X: 0.0555555555555556,
C: 0.138888888888889, D: 0.0185185185185185, J: 0.0555555555555556, E: 0.0277777777777778,
K: 0.0277777777777778, F: 0.00925925925925926, Q: 0.0370370370370370, L: 0.0370370370370370,
R: 0.0925925925925926, M: 0.0185185185185185, H: 0.0185185185185185, N: 0.120370370370370,
Y: 0.00925925925925926, O: 0.00925925925925926, U: 0.0555555555555556
```

Criptosistema de Hill

Como antes identificamos $\mathcal{A} \approx \mathbb{Z}_{26}$. Dada una matriz $M \in \mathcal{M}_t(\mathbb{Z}_{26})$ con inversa, la aplicación

$$e : \mathbb{Z}_{26}^t \to \mathbb{Z}_{26}^t, [e(x_0 \dots x_{t-1}) = (x_0 \dots x_{t-1})M]$$

es biyectiva, y proporciona una función de cifrado

$$e : \mathbb{Z}_{26}^* \to \mathbb{Z}_{26}^*,$$

definida dividiendo cada cadena en bloques de longitud t y multiplicando por M cada bloque.

Cifrado de Hill: Ejemplo I

```
sage: A = AlphabeticStrings()
sage: E = HillCryptosystem(A,4)
sage: K = random_matrix(IntegerModRing(26),4,4)
sage: while not(K.is_invertible()):
... K = random_matrix(IntegerModRing(26),4,4)
sage: P = A.encoding("Rugby union, or simply rugby, is a contact team sport which originated in England in the first
half of the 19th century. One of the two codes of rugby football, it is based on running with the ball in hand. In
its most common form, a game is between two teams of 15 players (two more than rugby league) using an oval-shaped
ball on a rectangular field with H-shaped goalposts on each try line. In 1845, the first football laws were written
by Rugby School pupils; other significant events in the early development of rugby include the Blackheath Club's
decision to leave the Football Association in 1863 and the split between rugby union and rugby league in 1895.
Historically an amateur sport, in 1995 restrictions on payments to players were removed, making the game openly
professional at the highest level for the first time.")
sage: C = E.enciphering(K,P); C
KAZERQLZDXAXCSXRZCVSHDLYIQYARBRVHJBFHWFNFJCAJCRRSEUFNODMXUTAFUROFXWYFTMKRLYNBWIVSJLWELAZIP
KRWMPGBEWWZGXAWYTDOQHXKAZELNZILHQQJCTIZDBWRHKDZQYZLTZYEJZCBUDKHXSCTCOXLPAPCSYEQQCPDOAMDFZC
CQAQPQHXYUSDVRTBAOSHPMAKCWIVNEPKEYBVURQDHRRPPMRWIISKUGTFBMTVJSTONMJGNTWYJUDQIOVSPFFGNGWNVU
SABASYTGLGTWHSDQSEVCBNZJRCWZFMHXSPPKOCTWXQTQIFFSYAQVPCCQYIZFNKVKKJKDQBKAZEQILVPCEGNPMDPKUM
VVDYDNMFKSVNHMDCNJJRIVSKNQOYZHINTTMPFICQQJILHDLYFQRSJCITQJOPHWXRYBIBDMREKMUVUYDTNCRGNNCNXE
BEQDARLVFNQUGOHRLCFSFQMPPZTYGKOPGZYUSDHRRPEYRILGJLKPMOJHAFICKEHUGAFMTLACAZXXDZMASBGMVQHFQZ
VIAFBGQURUFRFTRMBRBVFICQTHCLCWIVEGORNUBUTEHTKYGJJPKRGUEFMUDXUDSRZZMWUUCALPOXWZZSLLBSWCNUVS
TGNROCFTMKRLYNUIST
sage: E.deciphering(K,C)
RUGBYUNIONORSIMPLYRUGBYISACONTACTTEAMSPORTWHICHORIGINATEDINENGLANDINTHEFIRSTHALFOFTHETHCEN
TURYONEOFTHETWOCODESOFRUGBYFOOTBALLITISBASEDONRUNNINGWITHTHEBALLINHANDINITSMOSTCOMMONFORMA
```

Cifrado de Hill: Ejemplo II

```
GAMEISBETWEENTWOTEAMSOFPLAYERSTWOMORETHANRUGBYLEAGUEUSINGANOVALSHAPEDBALLONARECTANGULARFIE
LDWITHHSHAPEDGOALPOSTSONEACHTRYLINEINTHEFIRSTFOOTBALLLAWSWEREWRITTENBYRUGBYSCHOOLPUPILSOTH
ERSIGNIFICANTEVENTSINTHEEARLYDEVELOPMENTOFRUGBYINCLUDETHEBLACKHEATHCLUBSDECISIONTOLEAVETHE
FOOTBALLASSOCIATIONINANDTHESPLITBETWEENRUGBYUNIONANDRUGBYLEAGUEINHISTORICALLYANAMATEURSPOR
TINRESTRICTIONSONPAYMENTSTOPLAYERSWEREREMOVEDMAKINGTHEGAMEOPENLYPROFESSIONALATTHEHIGHESTLE
VELFORTHEFIRSTTIME
sage: C.frequency_distribution()
Discrete probability space defined by V: 0.0370370370370370,
X: 0.0277777777777778, J: 0.0308641975308642, L: 0.0354938271604938,
N: 0.0401234567901235, Y: 0.0370370370370370, P: 0.0401234567901235,
B: 0.0324074074074074, D: 0.0401234567901235, F: 0.0462962962962963,
Q: 0.0493827160493827, H: 0.0354938271604938, S: 0.0432098765432099,
U: 0.0401234567901235, W: 0.0324074074074074, I: 0.0370370370370370,
K: 0.0385802469135802, M: 0.0385802469135802, O: 0.0262345679012346,
Z: 0.0416666666666667, A: 0.0385802469135802, C: 0.0509259259259259,
E: 0.0308641975308642, G: 0.0324074074074074, R: 0.0540123456790123,
T: 0.0432098765432099
sage: aux = A('')
sage: for ii in range(len(C)):
... if ii% 4 == 0:
... aux *= C[ii]
...
sage: aux.frequency_distribution()
Discrete probability space defined by V: 0.0432098765432099,
```

Cifrado de Hill: Ejemplo III

X: 0.0185185185185185, J: 0.0432098765432099, L: 0.0493827160493827,
N: 0.0740740740740741, Y: 0.0185185185185185, P: 0.0432098765432099,
B: 0.0432098765432099, D: 0.0370370370370370, F: 0.0740740740740741,
Q: 0.0432098765432099, H: 0.0802469135802469, S: 0.0123456790123457,
U: 0.0370370370370370, W: 0.0308641975308642, I: 0.0370370370370370,
K: 0.0493827160493827, M: 0.0185185185185185, O: 0.0123456790123457,
Z: 0.0493827160493827, A: 0.0123456790123457, C: 0.0370370370370370,
E: 0.0308641975308642, G: 0.0123456790123457, R: 0.0370370370370370,
T: 0.0555555555555556

Criptosistema de permutación

Una permutación de t elementos $\sigma \in S_t$, es decir, una aplicación biyectiva $\sigma : \{0, \ldots, t-1\} \to \{0, \ldots, t-1\}$, induce otra biyección

$$e : \mathcal{A}^t \to \mathcal{A}^t, \left[e(x_0 \ldots x_{t-1}) = x_{\sigma(0)} \ldots x_{\sigma(t-1)} \right],$$

que se extiende de la manera usual a una función de cifrado

$$e : \mathcal{A}^* \to \mathcal{A}^*$$

descomponiendo cada cadena en bloques de tamaño t.

Cifrado de permutación: Ejemplo I

```
sage: A = AlphabeticStrings()
sage: E = TranspositionCryptosystem(A,10)
sage: K = E.random_key(); K
(1,9,6,8,5,10)(2,4)(3,7)
sage: P = A.encoding("Rugby union, or simply rugby, is a contact team sport which originated in England in the first
half of the 19th century. One of the two codes of rugby football, it is based on running with the ball in hand. In
its most common form, a game is between two teams of 15 players (two more than rugby league) using an oval-shaped
ball on a rectangular field with H-shaped goalposts on each try line. In 1845, the first football laws were written
by Rugby School pupils; other significant events in the early development of rugby include the Blackheath Club's
decision to leave the Football Association in 1863 and the split between rugby union and rugby league in 1895.
Historically an amateur sport, in 1995 restrictions on payments to players were removed, making the game openly
professional at the highest level for the first time.")
sage: while not(len(P)% 10 == 0):
... P *= A('X')
sage: len(P)
650
sage: C = E.enciphering(K,P); C
OBNUNIGYURRILRUYSMPONICBTOYSAGPTMCOSTEAARHHTIOWICRNADIEINTEGTAIGHNLNDNLRHFFAISTEEHHFNCTE
TOFYEUTORONTEWOESDTOCHOUYFOFRGBOSLTBBIALITNDRSNUEONAHWHNETGITINLHADALINBTTONCSISMIMOOMAR
MNFOTEBAWEMISGATTEMENWOERPYOSEFLASHMEWATOORTAGLRGEUBYNNSGEOAUINUDSPABELHAVCORLTELNAAIURN
EFGLAAHIHDASWTHLOGLESPDOAPTNCSRHOEATHNNLETIEIYTSOIBORTFFELSLRWLAWABIEWYNRTTEOBCUOHGYSRTP
SPHOUILLIIIRCFSGNESENNITTVEAYERTDLHEANNLMVTEEOPENUYFCIRGBOLEEUABDTHLLEHKUCHATCOESSNIDCIB
HEEOETLAVTATLOSLOBAFNIIOIOCATSPDEALSNTHNNEETREBTWINYIGAOBUNUEUYDALRGBNTIIUOSENHGNAYIAACL
LROESARPTURMIRTICRNESTANNIYPOSOTLTOEAPNSTMRSREEERWEYIEAONKVDMMOEMTPEHGAGEYONSFLPRETNAIHT
OALSLGSHETIHEEEFTEFHLORVXTMRXESTII
```

Cifrado de permutación: Ejemplo II

```
sage: E.deciphering(K,C)
RUGBYUNIONORSIMPLYRUGBYISACONTACTTEAMSPORTWHICHORIGINATEDINENGLANDINTHEFIRSTHALFOFTHETHC
ENTURYONEOFTHETWOCODESOFRUGBYFOOTBALLITISBASEDONRUNNINGWITHTHEBALLINHANDINITSMOSTCOMMONF
ORMAGAMEISBETWEENTWOTEAMSOFPLAYERSTWOMORETHANRUGBYLEAGUEUSINGANOVALSHAPEDBALLONARECTANGU
LARFIELDWITHHSHAPEDGOALPOSTSONEACHTRYLINEINTHEFIRSTFOOTBALLLAWSWEREWRITTENBYRUGBYSCHOOLP
UPILSOTHERSIGNIFICANTEVENTSINTHEEARLYDEVELOPMENTOFRUGBYINCLUDETHEBLACKHEATHCLUBSDECISION
TOLEAVETHEFOOTBALLASSOCIATIONINANDTHESPLITBETWEENRUGBYUNIONANDRUGBYLEAGUEINHISTORICALLYA
NAMATEURSPORTINRESTRICTIONSONPAYMENTSTOPLAYERSWEREREMOVEDMAKINGTHEGAMEOPENLYPROFESSIONAL
ATTHEHIGHESTLEVELFORTHEFIRSTTIMEXX
sage: P.frequency_distribution()
Discrete probability space defined by V: 0.00923076923076923,
X: 0.00307692307692308, L: 0.0584615384615384, N: 0.0784615384615385,
Y: 0.0276923076923077, P: 0.0230769230769231, B: 0.0292307692307692,
D: 0.0215384615384615, F: 0.0261538461538461, H: 0.0476923076923077,
S: 0.0584615384615384, U: 0.0307692307692308, W: 0.0184615384615385,
I: 0.0738461538461539, K: 0.00307692307692308, M: 0.0246153846153846,
O: 0.0784615384615385, A: 0.0753846153846154, C: 0.0261538461538461,
E: 0.106153846153846, G: 0.0323076923076923, R: 0.0553846153846154,
T: 0.0923076923076924
sage: C.frequency_distribution()
Discrete probability space defined by V: 0.00923076923076923,
X: 0.00307692307692308, L: 0.0584615384615384, N: 0.0784615384615385,
Y: 0.0276923076923077, P: 0.0230769230769231, B: 0.0292307692307692,
D: 0.0215384615384615, F: 0.0261538461538461, H: 0.0476923076923077,
```

Cifrado de permutación: Ejemplo III

S: 0.0584615384615384, U: 0.0307692307692308, W: 0.0184615384615385,
I: 0.0738461538461539, K: 0.00307692307692308, M: 0.0246153846153846,
O: 0.0784615384615385, A: 0.0753846153846154, C: 0.0261538461538461,
E: 0.106153846153846, G: 0.0323076923076923, R: 0.0553846153846154,
T: 0.0923076923076924

One-time-pad

Consiste en un cifrado de Vigenere con una clave "aleatoria" de la misma longitud, al menos, que el texto. Una versión eléctrica binaria fue desarrollada por Vernam y Mauborgne en la compañía AT&T. Matemáticamente, el mensaje y la clave son cadenas en $\mathbb{Z}_{26}^*$, siendo la clave una sucesión aleatoria. Si $m = m_0 m_1 \cdots$ y $k = k_0 k_1 \cdots$, la función de cifrado es

$$e(m) = (m_0 + k_0 \pmod{26})(m_1 + k_1 \pmod{26}) \cdots$$

Por qué un solo uso.

Si $m^{(1)} + k = c^{(1)}$ y $m^{(2)} + k = c^{(2)}$, $c^{(2)} - c^{(1)} = m^{(2)} - m^{(1)}$. Desaparece la clave y por tanto la aleatoriedad.

WWII

- Máquinas electromecánicas: Enigma, Purple, Typex, SIGABA,
- Libros de códigos.
- One-time-pad.

Índice

Definición formal I

Dados conjuntos

- $\mathcal{M}$ el conjunto se los mensajes, textos en claro o *plaintexts*,
- $\mathcal{C}$ el conjunto de los criptogramas o *cyphertexts*,
- $\mathcal{K}$ el espacio de claves o *key space*,

un cripsosistema viene definido por dos aplicaciones

$$e : \mathcal{K} \times \mathcal{M} \to \mathcal{C},$$
$$d : \mathcal{K} \times \mathcal{C} \to \mathcal{M},$$

tales que para cualquier clave $k \in \mathcal{K}$ y cualquier mensaje $m \in \mathcal{M}$,

$$d(k, e(k, m)) = m. \tag{1}$$

Definición formal II

Fijada una clave $k \in \mathcal{K}$, se suele utilizar la notación

$$e_k : \mathcal{M} \to \mathcal{C},$$
$$d_k : \mathcal{C} \to \mathcal{M},$$

para las funciones de cifrado y descifrado. La propiedad (1) se transforma en

$$d_k(e_k(m)) = m.$$

Consideraciones

Si partimos de un texto cifrado, ¿cómo es posible asegurarnos de que nuestro criptosistema es seguro? Vamos a hacer dos consideraciones de cara a realizar un pequeño estudio de la seguridad de un criptosistema

- *"el criptosistema no debe dar más información que la estrictamente necesaria"*.
- *"cada posible texto cifrado y cada posible texto sin cifrar son equiprobables"*.

Si tenemos en cuenta estas dos reglas nos aseguramos de que nuestro criptosistema no contiene información redundante, y por tanto todo criptoanálisis que se realice se debe centrar únicamente en los métodos matemáticos usados en su diseño o en la potencia de cálculo que se pueda desarrollar.

Aproximación de Shannon

En su artículo "Communication Theory of Secrecy Systems", C. E. Shannon destaca dos características que un criptosistema debe tener para no ser vulnerable a ataques estadísticos y de frecuencias:

Difusión La estructura estadística del mensaje, que produce su redundancia, se disipa en la estructura estadística de grandes combinaciones de letras del criptograma. Informalmente, un cambio en un carácter del texto en claro provocará muchos cambios en el criptograma (idealmente en la mitad sus caracteres).

Confusión La relación entre el criptograma y la clave es compleja y enmarañada. Esto quiere decir que cada carácter del criptograma depende de varios caracteres de la clave.

Desde su publicación, se ha buscado garantizar estas características en los criptosistemas propuestos.

Lo importante, desde nuestro punto de vista, es que existen herramientas para analizar la bondad de un criptosistema.

Criptografía digital

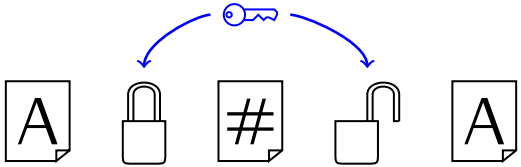
Con el desarrollo de la sociedad de la información y la digitalización de los contenidos, los criptosistemas modernos se diseñan sobre cadenas de bits, y no sobre cadenas de caracteres. Fijamos en consecuencia la siguiente notación. $\mathbb{B} = \mathbb{F}_2 = \mathbb{Z}_2$ es el álgebra de Boole con dos elementos, y en él tenemos las operaciones $\{\wedge, \vee, \oplus, \cdot, +\}$.

Los mensajes, criptogramas y claves son cadenas de bits, es decir, $\mathcal{M} = \mathcal{C} = \mathcal{K} = \mathbb{B}^*$.

Criptosistemas simétricos

Las claves de cifrado y descifrado son "computacionalmente" equivalentes, es decir, si conocemos la clave utilizada para cifrar el mensaje, podemos descifrar el criptograma. Todos los criptosistemas clásicos son simétricos. En la actualidad los hay de dos clases:

1. Cifrados de bloque.
2. Cifrados de flujo.



Índice

Definición

Para un criptosistema por bloques, limitamos los mensajes, criptogramas y claves a cadenas de una longitud fija,

$$e : \mathbb{B}^K \times \mathbb{B}^N \to \mathbb{B}^M,$$
$$d : \mathbb{B}^K \times \mathbb{B}^M \to \mathbb{B}^N,$$

o para cada clave $k \in \mathbb{B}^K$,

$$e_k : \mathbb{B}^N \to \mathbb{B}^M,$$
$$d_k : \mathbb{B}^M \to \mathbb{B}^N.$$

Lo usual es que $N = M$, hipótesis que asumiremos mientras no indiquemos lo contrario. A este valor común se le conoce como tamaño del bloque y a K como el tamaño de la clave.

Cómo se extiende e de $\mathbb{B}^N$ a $\mathbb{B}^*$ es competencia de los modos de operación. Estos modos dependen del tamaño del bloque, no de la clave.

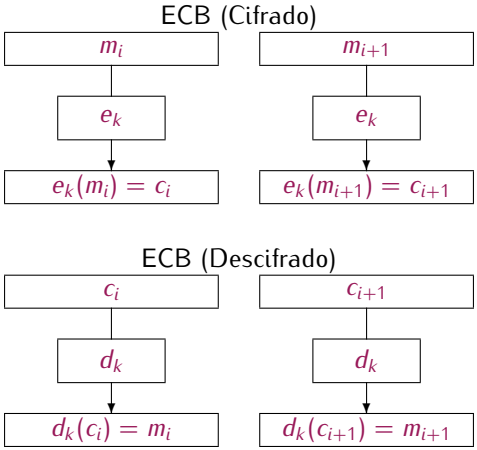
Electronic Code Book (ECB)

El mensaje se divide en bloques de N bits, es decir, $m = m_0||m_1||\cdots$ donde cada m_i tiene longitud N. Cada bloque se cifra de manera independiente. Formalmente,

$$c_i = e_k(m_i)$$

Difusión y confusión se mantienen localmente en cada bloque. Un atacante podría sustituir un bloque del criptograma por otro de texto en claro conocido.

ECB: descripción gráfica



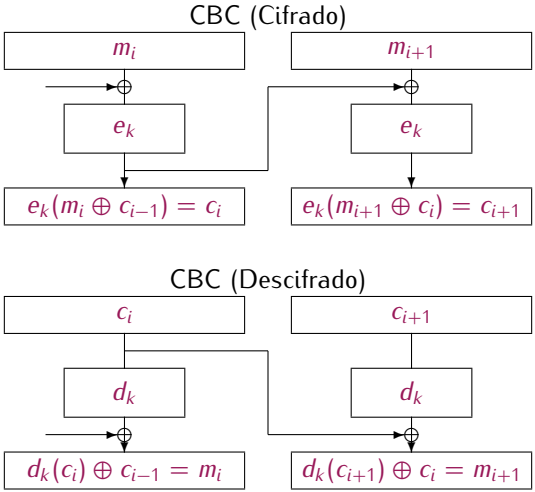
Cipher Block Chaining (CBC)

En este modo, la salida del cifrado de un bloque se suma (XOR) a la entrada del siguiente bloque:

$$c_{i+1} = e(m_{i+1} \oplus c_i)$$

Empleamos un vector de inicialización $c_0 = \texttt{IV}$ que no es necesario mantener en secreto. Su integridad es esencial para garantizar el correcto descifrado de c_1. Si `IV` es aleatorio, un mismo texto en claro corresponderá con diferentes criptogramas en cada cifrado.

CBC: descripción gráfica



Cipher FeedBack (CFB)

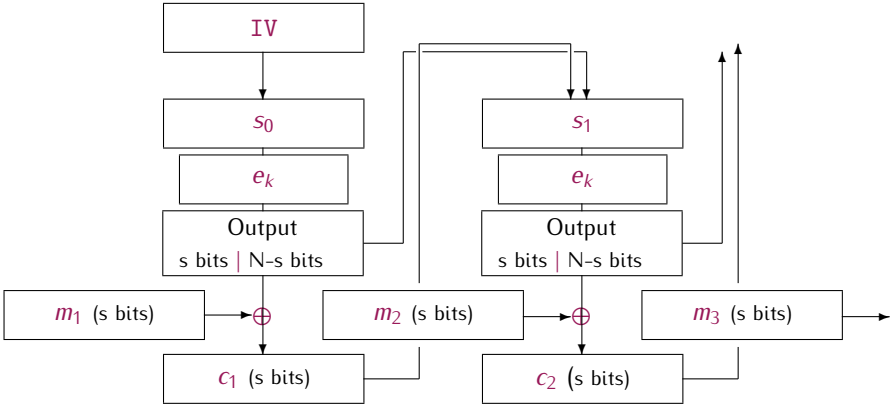
- Recordemos que N es el tamaño del bloque. Sea s un entero tal que $1 \leq s \leq N$.
- El mensaje se divide en bloques de tamaño s, que llamamos m_i, $i \geq 1$.
- Tomamos un $s_0 = \texttt{IV}$ de longitud N. Las funciones LSB_{N-s} y MSB_s devuelven los $N-s$ bits menos significativos (por la derecha) y los s más significativos (por la izquierda).
- Una vez que hemos generado un bloque s_{i-1} lo ciframos. La parte correspondiente del criptograma es $c_i := m_i \oplus \mathrm{MSB}_s(e_k(s_{i-1}))$. Además $s_i = (\mathrm{LSB}_{N-s}(s_{i-1}), c_i)$.

Es decir,

$$
\begin{aligned}
& s_0 \\
& c_1 := m_1 \oplus \mathrm{MSB}_s(e_k(s_0)) \\
& s_1 := (\mathrm{LSB}_{N-s}(e_k(s_0)), c_1) \\
& c_i := m_i \oplus \mathrm{MSB}_s(e_k(s_{i-1})) \quad \text{for } i = 2, \ldots \\
& s_i := (\mathrm{LSB}_{N-s}(e_k(s_{i-1})), c_i) \quad \text{for } i = 2, \ldots.
\end{aligned}
$$

Cuando $s = N$, MSB_s es la identidad y LSB_{N-s} es el bloque vacío.

CFB: descripción gráfica del cifrado



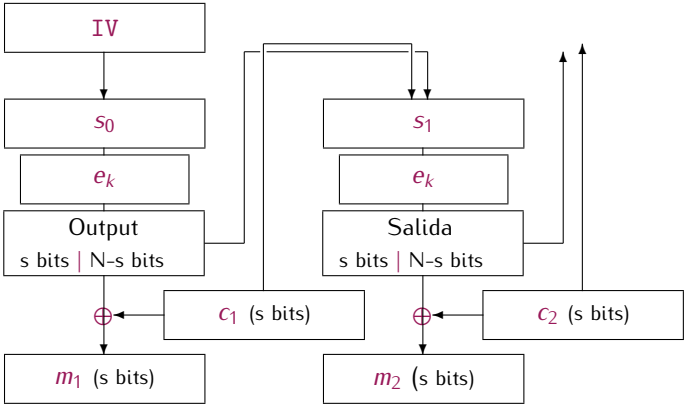
CFB: descifrado

- Se emplea el mismo vector de inicialización $s_0 = \texttt{IV}$.
- Una vez calculado s_{i-1}, podemos calcular el siguiente bloque del mensaje mediante $m_i := c_i \oplus \mathrm{MSB}_s(e_k(s_0))$.
- De nuevo $s_i = (\mathrm{LSB}_{N-s}(s_{i-1}), c_i)$.

Formalmente:

$$
\begin{aligned}
& s_0 \\
& m_1 := c_1 \oplus \mathrm{MSB}_s(e_k(s_0)) \\
& s_1 := (\mathrm{LSB}_{b-s}(e_k(s_0)), c_1) \\
& m_i := c_i \oplus \mathrm{MSB}_s(e_k(s_{i-1})) \quad \text{for } i = 2, \ldots, n \\
& s_i := (\mathrm{LSB}_{N-s}(e_k(s_{i-1})), c_i) \quad \text{for } i = 2, \ldots, n.
\end{aligned}
$$

CFB: descripción gráfica del descifrado

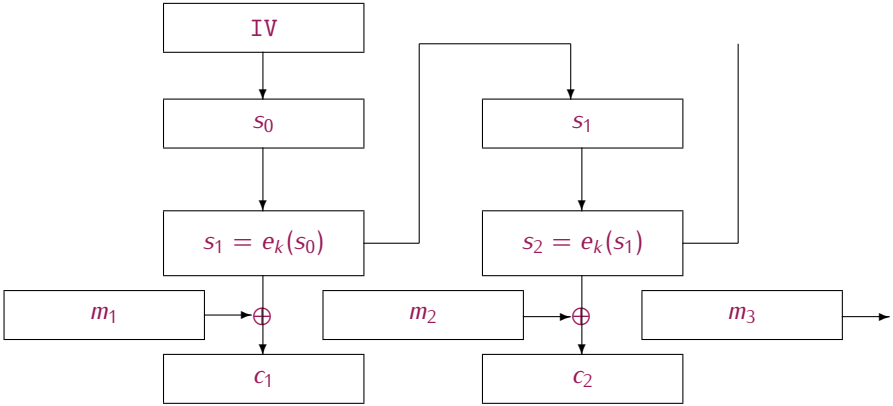


Output FeedBack (OFB)

Partimos de nuevo de un vector de inicialización $s_0 = \texttt{IV}$. El cifrado por bloques actúa como la función generadora de un cifrado de flujo síncrono, sistema que estudiaremos más adelante. Formalmente:

$$
\begin{aligned}
& s_0 \\
& c_i := m_i \oplus e_k(s_{i-1}) \quad \text{for } i = 1, \ldots, n \\
& s_i := e_k(s_{i-1}) \quad \text{if } i = 1, \ldots, n.
\end{aligned}
$$

OFB: descripción gráfica del cifrado

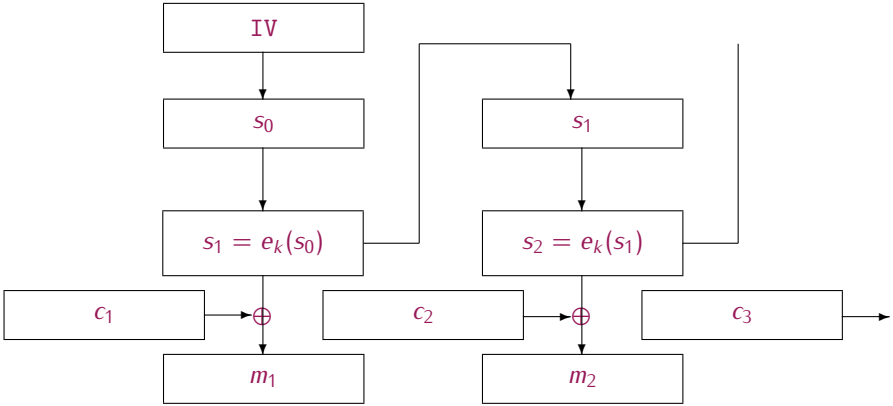


OFB: descifrado

El proceso para descifrar es exactamente el mismo, empleando el mismo vector de inicialización:

$$
\begin{aligned}
& s_0 \\
& m_i := c_i \oplus e_k(s_{i-1}) \quad \text{for } i = 1, \ldots, n \\
& s_i := e_k(s_{i-1}) \quad \text{if } i = 1, \ldots, n.
\end{aligned}
$$

OFB: descripción gráfica del descifrado



Counter Mode (CTR)

Este modo parte de un vector inicial IV, llamado aquí valor de un solo uso (*nonce* en inglés), y una función contador que genera una sucesión

$$h_0 = \mathtt{IV}, \ldots, h_{i+1} = \mathrm{CTR}(h_i), \ldots, h_t$$

de manera que si el mensaje se descompone en bloques como $m = m_0||m_1||\cdots||m_t$ el cifrado se produce de la forma

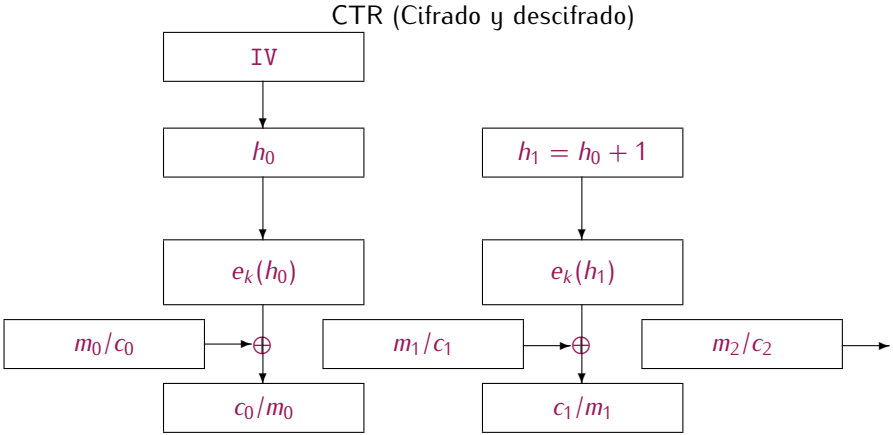
$$c_i = m_i \oplus e_k(h_i) \quad \text{para } i = 0, \ldots, t.$$

El descifrado actúa de la misma manera.

$$m_i = c_i \oplus e_k(h_i) \quad \text{para } i = 1, \ldots, t.$$

Es otra forma de utilizar el criptosistema por bloques como un cifrado de flujo.
La función contador debe garantizar que no se emplea el mismo valor para cifrar dos bloques diferentes. Suele emplearse el incremento en una unidad, es decir, $\mathrm{CTR}(h_{i+1}) = h_i + 1$.

CTR: descripción gráfica



Índice

Red de Feistel: Cifrado

El proceso de cifrado funciona de la siguiente forma:

1. La entrada es un bloque de longitud N.
2. Este bloque se divide en dos partes de longitud $N/2$, llamadas L_0 y R_0.
3. Durante n rondas, es decir, moviendo i de 1 a n,

$$L_i = R_{i-1} \qquad R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

   donde K_i es la subclave usada en cada ronda.
4. La salida es el bloque $R_n||L_n$.

Red de Feistel: Descifrado

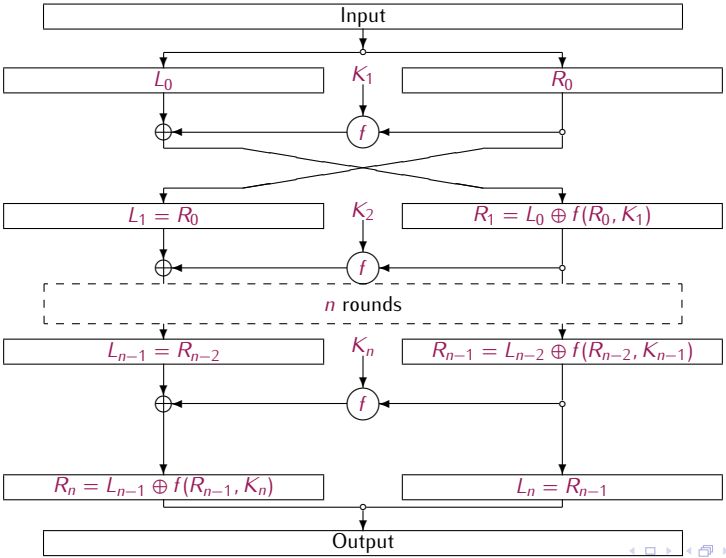
El proceso de descifrado es simétrico

1. La entrada es un bloque de longitud N.
2. Este bloque se divide en dos de longitud $N/2$, llamados R_n y L_n
3. Para cada i entre $n-1$ y 0,

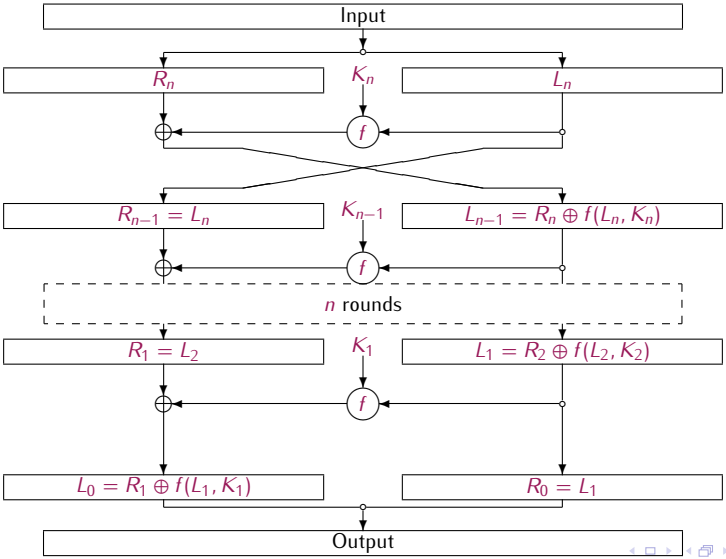
$$R_i = L_{i+1} \qquad L_i = R_{i+1} \oplus f(L_{i+1}, K_{i+1})$$

   donde K_i es la subclave usada en cada ronda.
4. La salida es el bloque $L_0||R_0$.

Red de Feistel: Descripción gráfica del cifrado



Red de Feistel: Descripción gráfica del descifrado



DES: Punto de partida

DES fue un desarrollo realizado a partir de una solicitud del organismo americano entonces llamado NBS (National Boureau of Standards) —hoy rebautizado como NIST (National Institute of Standards and Technology)— por IBM, basado en un criptosistema previo llamado LUCIFER. Debía satisfacer los siguientes requisitos:

- El algoritmo debe proporcionar un alto nivel de seguridad.
- Debe ser fácil de describir y completamente especificado.
- La seguridad debe recaer en la clave, no en el secreto del algoritmo.
- Totalmente accesible a todos los usuarios.
- Fácil de adaptar a diferentes aplicaciones.
- Fácil de implementar en hardware.
- Debe ser eficiente..

DES: Observaciones sobre su seguridad

1. La longitud de la clave, 56 bits, fue considerada pequeña demasiado pronto.
2. Los criterios de diseño de las S-cajas fue mantenido en secreto hasta 1994, lo que hizo creer a mucha gente que tenían una puerta secreta.

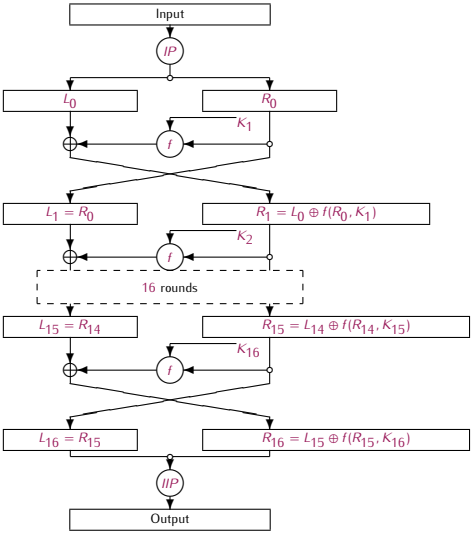
Hoy en día una clave puede obtenerse en 24 horas.

DES: Descripción

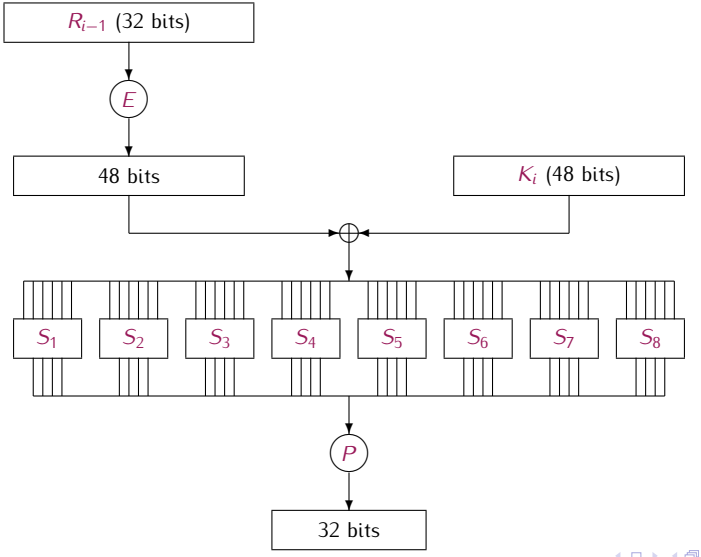
DES es un cifrado por bloques.

1. La entrada es un bloque de 64 bits, y la salida tiene la misma longitud.
2. DES es una red de Feistel de 16 rondas.
3. Cifrado y descifrado son idénticos excepto en el orden de las subclaves.
4. La clave tiene una longitud de 56 bits, aunque se presenta como un bloque de 64 donde los bits 8, 16, 24, 32, 40, 48, 56 y 64 son bits de paridad.
5. Hay claves débiles y semidébiles, pero son conocidas y fácilmente eliminadas.
6. Las primeras implementaciones fueron en hardware. No es muy rápido en sus versiones software.

DES: Descripción gráfica del cifrado



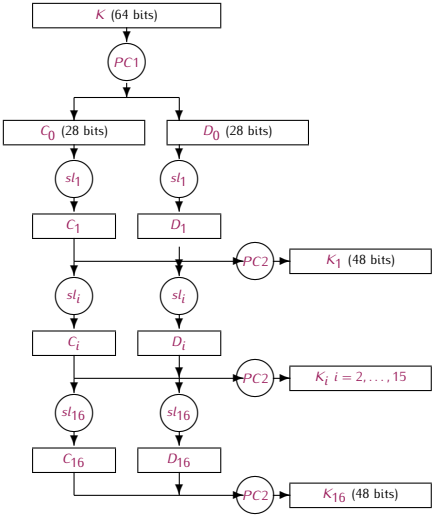
DES: Función f



DES: Expansión de la clave

1. Los bits de paridad se desechan y se permutan los demás bits mediante una función llamada *Permutation Choice* 1 $PC1$.
2. Este bloque de 56 bits se divide en dos bloque de 28 bits. Si $K = \langle k_1, \ldots, k_{28}, k_{29}, \ldots, k_{56} \rangle$, llamamos $C = \langle k_1, \ldots, k_{28} \rangle$ and $D = \langle k_{29}, \ldots, k_{56} \rangle$.
3. Realizamos un desplazamiento cíclico a la izquierda de una o dos posiciones, dependiendo de la ronda, en los bloques C y D.
4. En cada ronda seleccionamos 48 bits usando la *Permutation Choice 2* $PC2$.

DES: Descripción gráfica de la expansión de la clave



DES: Expansión y permutación E

Esta función transforma un bloque de longitud 32 en otro de 48 bits duplicando la mitad de ellos. Todos son reordenados.

- Los 48 bits son sumados a la clave de ronda.
- Los bits duplicados serán comprimidos más adelante.
- Esta técnica aumenta la aleatoriedad entre la entrada y la salida.

S-cajas

- Las funciones de sustitución empleadas son 8 llamadas S-cajas.
- Cada bloque de 48 bits se divide en 8 subbloques de 6 bits, siendo operado cada uno de ellos por una S-caja distinta.
- Las S- cajas se representan por una matriz 4×16, con filas numeradas de 0 a 3 y columnas de 0 a 15.
- De cada entrada $\langle b_0 b_1 b_2 b_3 b_4 b_5 \rangle$ ontenemos dos números que representan una fila y una columna. La fila es $r = b_0 2 + b_5$ y la columna $c = b_1 2^3 + b_2 2^2 + b_3 2 + b_4$.
- El valor en binario correspondiente a la posición $\langle r, c \rangle$ nos da una salida de 4 bits.
- La concatenación de las 8 salidas nos da un nuevo bloque de longitud 32.

Claves débiles

En 1992 Campbell y Wiener demostraron que DES está muy lejos de ser un grupo, es decir, existen claves K^1 y K^2 tales que para cualquier otra clave K^3

$$\mathrm{DES}_{K^2} \circ \mathrm{DES}_{K^1} \neq \mathrm{DES}_{K^3}.$$

Esta falta de estructura hace que el criptoanálisis sea más difícil.

Una clave es *débil* si todas las subclaves de ronda son iguales. Es *semidébil* si solo genera dos o cuatro claves de ronda diferentes y alternadas. Hay 4 claves débiles y 60 (12 con dos claves de ronda y 48 cuatro) claves semidébiles. Todas están tabuladas y no representan un problema de seguridad por su pequeña cantidad. La débiles son en hexadecimal

0x0101010101010101
0x1F1F1F1F0E0E0E0E
0xE0E0E0E0F1F1F1F1
0xFEFEFEFEFEFEFEFE

Índice

AES: Introducción

Es el nuevo estándar diseñado para reemplazar a DES. Se empieza a gestar el 12 de septiembre de 1997, cuando el departamento de comercio del *National Institute of Standards and Technology (NIST)* hace un llamamiento público para la presentación de algoritmos.
Los requisitos mínimos son:

1. El algoritmo debe ser simétrico de clave secreta.
2. El algoritmo debe ser un algoritmo de bloque.
3. El algoritmo debe ser capaz de soportar las combinaciones clave-bloque de tamaños 128-128, 192-128 y 256-128.

AES: Evaluación

Los criterios de evaluación de los proyectos presentados fueron los siguientes:

Seguridad. El factor más importante en la evaluación de los candidatos.

Coste.

1. El algoritmo debe ser accesible a todo el mundo y de libre distribución.
2. El algoritmo debe ser computacionalmente eficiente tanto en hardware como en software.
3. El algoritmo debe utilizar la menor memoria posible tanto en hardware como en software.

AES: Características de implementación del algoritmo.

1. El algoritmo debe ser fácilmente implementable en distintas plataformas tanto en hardware como en software.
2. El algoritmo debe acomodarse a diferentes combinaciones clave-bloque además de las mínimas requeridas.
3. El algoritmo debe ser de diseño simple.

AES: candidatos

A modo de curiosidad veamos algunos de los candidatos que se presentaron y sus promotores.

CAST-256	Entust Technologies, Inc. (C. Adams).
CRYPTON	Future Systems, Inc. (Chae Hoon Lim).
DEAL	L. Knudsen, R. Outerbridge.
DFC	CNRS-Ecole Normale Superiere (S. Vaudenay).
E2	NTT Nippon Telegraph and Telephone Corporation (M. Kanda).
FROG	TecApro International S.A. (D. Georgoudis, D. Leroux, B. S. Chaves).
HPC	R. Schoeppel.
LOKI97	L. Brown, J. Pieprzyk, J.Seberry.
MAGENTA	Deutshe Telekom AG (K. Huber).
MARS	IBM (N. Zunic).
RC6	RSA Laboratories (Rivest, M. Robshaw, Sidney, Yin).
RIJNDAEL	J. Daemen, V. Rijmen.
SAFER+	Cylink Corporation (L. Chen).
SERPENT	R. Anderson, E. Biham, L. Knudsen.
TWOFISH	B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson.

AES: RIJNDAEL

- Es un algoritmo simétrico de bloque de 128 bits y clave de 128, 192 o 256 bits.
- Para representar los bloques usamos polinomios de grado tres con coeficientes en $\mathbb{F}_{2^8} = \mathbb{F}_{256}$, esto es el cuerpo finito de 256 elementos.
- La representación del cuerpo $\mathbb{F}_{256}$ se realiza mediante el polinomio $x^8 + x^4 + x^3 + x + 1$, que es un polinomio irreducible sobre el cuerpo $\mathbb{F}_2$, el cuerpo finito de dos elementos.
- El grupo multiplicativo de $\mathbb{F}_{256}$ es un grupo cíclico, generado por la clase de $x + 1$. Esto es, los elementos de $\mathbb{F}_{256}$ se pueden escribir como potencias de $x + 1$.

AES: elementos de $\mathbb{F}_{256}$

Cada elemento de $\mathbb{F}_{256}$ es la clase de un polinomio de grado menor que ocho, por tanto, escribiéndolo como una lista de sus coeficientes, sería

$$a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 \leftrightarrow a_7a_6a_5a_4a_3a_2a_1a_0.$$

Esto es, una 8-upla de elementos de $\mathbb{F}_2 = \{0, 1\}$.

Se puede identificar pues con un *byte*. Los bytes, se pueden escribir bien en modo hexadecimal o binario. De esta forma tenemos la correspondencia

byte	$\rightarrow$	hexadecimal	$\rightarrow$	polinomio
10100101	$\rightarrow$	0xA5	$\rightarrow$	$x^7 + x^5 + x^2 + 1$.

AES: elementos de $\mathbb{F}_{256}$ no nulos

El grupo multiplicativo de $\mathbb{F}_{256}$ es un grupo cíclico, de orden 255. De entre sus generadores el, posiblemente, más sencillo es el 0x03, que corresponde al polinomio $x + 1$. Todas las potencias de 0x03, escritas en hexadecimal, son:

03	05	0F	11	33	55	FF	1A	2E	72	96	A1	F8	13	35	5F	16
E1	38	48	D8	73	95	A4	F7	02	06	0A	1E	22	66	AA	E5	32
34	5C	E4	37	59	EB	26	6A	BE	D9	70	90	AB	E6	31	53	48
F5	04	0C	14	3C	44	CC	4F	D1	68	B8	D3	6E	B2	CD	4C	64
D4	67	A9	E0	3B	4D	D7	62	A6	F1	08	18	28	78	88	83	80
9E	B9	D0	6B	BD	DC	7F	81	98	B3	CE	49	DB	76	9A	B5	96
C4	57	F9	10	30	50	F0	0B	1D	27	69	BB	D6	61	A3	FE	112
19	2B	7D	87	92	AD	EC	2F	71	93	AE	E9	20	60	A0	FB	128
16	3A	4E	D2	6D	B7	C2	5D	E7	32	56	FA	15	3F	41	C3	144
5E	E2	3D	47	C9	40	C0	5B	ED	2C	74	9C	BF	DA	75	9F	160
BA	D5	64	AC	EF	2A	7E	82	9D	BC	DF	7A	8E	89	80	9B	176
B6	C1	58	E8	23	65	AF	EA	25	6F	B1	C8	43	C5	54	FC	192
1F	21	63	A5	F4	07	09	1B	2D	77	99	B0	CB	46	CA	45	208
CF	4A	DE	79	8B	86	91	A8	E3	3E	42	C6	51	F3	0E	12	224
36	5A	EE	29	7B	8D	8C	8F	8A	85	94	A7	F2	0D	17	39	240
4B	DD	7C	84	97	A2	FD	1C	24	6C	B4	C7	52	F6	01		255

AES: suma

Es un XOR a nivel de bits. Por ejemplo

$$(x^4 + x^3 + 1) + (x^7 + x^6 + x^4 + x^2 + 1) = x^7 + x^6 + x^3 + x^2$$

$$00011001 \oplus 11010101 = 11001100$$

$$0x19 \oplus 0xD5 = 0xCC$$

AES: multiplicación

Producto de polinomios y reducción módulo $x^8 + x^4 + x^3 + x + 1$. Por ejemplo

$$\begin{aligned}(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) \\ &= (x^{13} + x^{11} + x^9 + x^8 + x^7) + (x^7 + x^5 + x^3 + x^2 + x) + (x^6 + x^4 + x^2 + x + 1) \\ &= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \\ &\equiv x^7 + x^6 + 1 \pmod{x^8 + x^4 + x^3 + x + 1}.\end{aligned}$$

En hexadecimal y a nivel de bits

$$0x57 \bullet 0x83 = 0xC1 \qquad 01010111 \bullet 10000011 = 11000001$$

¿Cómo hacer esto de forma eficiente?

AES: multiplicación por recurrencia

Para hacer la multiplicación basta multiplicar por x, y después hacer la suma, XOR. El producto $(x^6 + x^4 + x^2 + x + 1) \cdot (x^7 + x + 1)$ es la suma

$$(x^6 + x^4 + x^2 + x + 1) \cdot x^7 + (x^6 + x^4 + x^2 + x + 1) \cdot x + (x^6 + x^4 + x^2 + x + 1)$$

También se puede realizar como:

$$(x^6 + x^4 + x^2 + x + 1) \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x \cdot x + (x^6 + x^4 + x^2 + x + 1) \cdot x + (x^6 + x^4 + x^2 + x + 1)$$

y vamos haciendo por recurrencia las multiplicaciones.

AES: multiplicación con logaritmos

Recordemos que los elementos no nulos de $\mathbb{F}_{256}$ forman un grupo cíclico de orden 255 generado por $x+1 = 0\text{x}03 = 00000011$. Esto significa que todo elemento distinto de cero es una potencia de $x+1$, por ejemplo

$$11001100 = 0\text{xAA} = x^7 + x^6 + x^3 + x^2 = (x+1)^{31},$$

por lo que

$$\log_{0\text{x}03} 0\text{xAA} = 31.$$

La multiplicación se realiza empleando tablas de logaritmos. Así, si $a, b \in \mathbb{F}_{256}^*$

$$ab = 0\text{x}03^{\log_{0\text{x}03} a + \log_{0\text{x}03} b \pmod{255}}.$$

Por ejemplo

$$0\text{x}57 \bullet 0\text{x}83 = 0\text{x}03^{\log_{0\text{x}03} 0\text{x}57 + \log_{0\text{x}03} 0\text{x}83 \pmod{255}} = 0\text{x}03^{98+80 \pmod{255}} = 0\text{x}03^{178} = 0\text{xC}1.$$

Las tablas de logaritmos se tabulan.

AES: multiplicación extendida I

Para algunos procesos AES emplea subbloques de 32 bits, es decir, necesitamos extender la aritmética a bloques de 4 bytes. Podemos representar estos bloque como polinomios de grado menor o igual que 3 con coeficientes en $\mathbb{F}_{256}$.
La suma sigue realizándose a nivel de bits mediante XOR. Vamos a multiplicar dos polinomios de grado menor que 4 con coeficientes en $\mathbb{F}_{256}$, si $a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ y $b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$, entonces el producto es un polinomio de hasta grado 6. Sea $c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$, siendo

$$
\begin{aligned}
c_0 &= a_0 \bullet b_0 \\
c_1 &= a_1 \bullet b_0 \oplus a_0 \bullet b_1 \\
c_2 &= a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2 \\
c_3 &= a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3 \\
c_4 &= a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3 \\
c_5 &= a_3 \bullet b_2 \oplus a_2 \bullet b_3 \\
c_6 &= a_3 \bullet b_3
\end{aligned}
$$

AES: multiplicación extendida II

para tener un polinomio de grado menor o igual que 3 basta reducir módulo un polinomio de grado 4. En este caso el polinomio $x^4 + 1$. Resulta entonces la relación:

$$x^h \equiv x^{h \pmod 4} \quad (\text{mód } x^4 + 1).$$

Al aplicar esto a la relación anterior resulta que tenemos que sumar c_0 y c_4, c_1 y c_5 y c_2 y c_6. Si llamamos

$$d(x) = a(x)b(x) \quad (\text{mód } x^4 + 1),$$

y si $d(x) = d_3x^3 + d_2x^2 + d_1x + d_0$, tenemos:

$$\begin{aligned} d_0 &= a_0 \bullet b_0 \oplus a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3 \\ d_1 &= a_1 \bullet b_0 \oplus a_0 \bullet b_1 \oplus a_3 \bullet b_2 \oplus a_2 \bullet b_3 \\ d_2 &= a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2 \oplus a_3 \bullet b_3 \\ d_3 &= a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3, \end{aligned}$$

AES: multiplicación extendida III

Esta multiplicación admite una representación matricial

$$\begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Esta transformación se puede invertir cuando $a(x)$ es un polinomio invertible módulo $x^4 + 1$.
(Observemos que $x^4 + 1$ no es irreducible; y que necesitamos que $a(x)$ sea primo relativo con $x^4 + 1$.)
En el caso particular en que $a(x) = x$, que es evidentemente primo relativo con $x^4 + 1$, la transformación es:

$$\begin{pmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{pmatrix} 00 & 00 & 00 & 01 \\ 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 00 & 00 & 01 & 00 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} b_3 \\ b_0 \\ b_1 \\ b_2 \end{pmatrix}$$

Tenemos entonces una permutación que mueve los elementos una unidad a la derecha.

Parámetros

Tenemos tres números que determinan el funcionamiento del criptosistema, estos son:

- N_b es el número de bits del bloque dividido por 32. En Rijndael sus valores son: 4, 6 u 8 según que el bloque tenga longitud 128, 192 o 256. El estándar AES sólo admite bloques de tamaño 128, por lo que el valor es 4
- N_k es el número de bits de la clave dividido por 32. Sus valores son: 4, 6 u 8 según que la clave tenga longitud 128, 192 o 256.
- N_r el *número de rondas* determinado por los dos anteriores según el cuadro:

N_r	$N_b = 4$	$N_b = 6$	$N_b = 8$
$N_k = 4$	**10**	12	14
$N_k = 6$	**12**	12	14
$N_k = 8$	**14**	14	14

Estado o *state*

Se actúa sobre un resultado intermedio, al que llamamos *state*. Éste es una estructura del siguiente tipo:

$m_{0,0}$	$m_{0,1}$	$m_{0,2}$	$m_{0,3}$	$\cdots$	$\cdots$
$m_{1,0}$	$m_{1,1}$	$m_{1,2}$	$m_{1,3}$	$\cdots$	$\cdots$
$m_{2,0}$	$m_{2,1}$	$m_{2,2}$	$m_{2,3}$	$\cdots$	$\cdots$
$m_{3,0}$	$m_{3,1}$	$m_{3,2}$	$m_{3,3}$	$\cdots$	$\cdots$

Donde cada $m_{i,j}$ son 8 bits, esto es, un byte, o si se quiere un elemento de $\mathbb{F}_{256}$.
El número de columnas es N_b, esto es, 4, 6 u 8 según el tamaño del bloque.

Primer estado

El primer *state* se forma, por columnas, dividiendo el bloque (por ejemplo de 128 bits, o equivalentemente 16 bytes)

$$m_{0,0}m_{1,0}m_{2,0}m_{3,0}m_{0,1}m_{1,1}m_{2,1}m_{3,1}m_{0,2}m_{1,2}m_{2,2}m_{3,2}m_{0,3}m_{1,3}m_{2,3}m_{3,3}$$

en los subbloques correspondientes según en siguiente esquema:

$m_{0,0}$	$m_{0,1}$	$m_{0,2}$	$m_{0,3}$	$\cdots$	$\cdots$
$m_{1,0}$	$m_{1,1}$	$m_{1,2}$	$m_{1,3}$	$\cdots$	$\cdots$
$m_{2,0}$	$m_{2,1}$	$m_{2,2}$	$m_{2,3}$	$\cdots$	$\cdots$
$m_{3,0}$	$m_{3,1}$	$m_{3,2}$	$m_{3,3}$	$\cdots$	$\cdots$

Rondas

La primera ronda, ronda 0, consiste en

1. $\texttt{AddRoundKey}(state, RoundKey_0)$

A continuación se realizan $N_r - 1$ rondas con el siguiente esquema: $\texttt{Round}(state, RoundKey_i)$, $i = 1, \ldots, N_r - 1$

1. $\texttt{SubBytes}(state)$
2. $\texttt{ShiftRows}(state)$
3. $\texttt{MixColumns}(state)$
4. $\texttt{AddRoundKey}(state, RoundKey_i)$

Y finalmente la última ronda con el esquema: $\texttt{FinalRound}(state, RoundKey_{N_r})$

1. $\texttt{SubBytes}(state)$
2. $\texttt{ShiftRows}(state)$
3. $\texttt{AddRoundKey}(state, RoundKey_{N_r})$

AddRoundKey

La función

$$\texttt{AddRoundKey}(state, RoundKey_i),$$

consiste en un XOR entre $state$ y $RounKey_i$ bit a bit; ($\oplus$ en la notación anterior).

$$state \oplus RounKey_0$$

SubBytes

Cada uno de los $m_{i,j}$ es un byte (8 bits), entonces:

1. Se calcula el inverso de $m_{i,j}$ en $\mathbb{F}_{256}$; la imagen de 00 es él mismo, obteniendo otro byte (8 bits) $(x_0 \cdots x_7)$,
2. A este byte se le aplica la transformación afín (invertible) definida por:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

en donde la columna de la derecha es el número 0x63, (hexadecimal).

ShiftRows

Las filas de *state* se desplazan a la izquierda cíclicamente.
La primera fila permanece en su posición, la segunda se desplaza C1 posiciones a la izquierda, la tercera C2 y la cuarta C3, según la siguiente tabla que depende de N_b.

Nb	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	4

Por ejemplo en el caso de $N_b = 4$ tenemos:

$m_{0,0}$	$m_{0,1}$	$m_{0,2}$	$m_{0,3}$
$m_{1,0}$	$m_{1,1}$	$m_{1,2}$	$m_{1,3}$
$m_{2,0}$	$m_{2,1}$	$m_{2,2}$	$m_{2,3}$
$m_{3,0}$	$m_{3,1}$	$m_{3,2}$	$m_{3,3}$

$\rightsquigarrow$

$m_{0,0}$	$m_{0,1}$	$m_{0,2}$	$m_{0,3}$
$m_{1,1}$	$m_{1,2}$	$m_{1,3}$	$m_{1,0}$
$m_{2,2}$	$m_{2,3}$	$m_{2,0}$	$m_{2,1}$
$m_{3,3}$	$m_{3,0}$	$m_{3,1}$	$m_{3,2}$

MixColumns

Las columnas de *state* se consideran polinomios, hasta grado tres, sobre $\mathbb{F}_{256}$ y se multiplican, módulo el polinomio $x^4 + 1$, por el polinomio con coeficientes en $\mathbb{F}_{256}$:
$c(x) = \texttt{0x03}x^3 + \texttt{0x01}x^2 + \texttt{0x01}x + \texttt{0x02}$, (los coeficientes están escritos en hexadecimal).
Como esta operación es siempre la misma, podemos dar una fórmula genérica. Se tiene que el producto $b(x) = c(x) \otimes a(x)$ está dado por la multiplicación matricial:

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Formación de las claves de ronda

- La clave *Key* se extiende a una lista de palabras de 4 bytes que llamaremos **W** y que contiene exactamente $N_b(N_r + 1)$ palabras.
- Los primeros N_k elementos de **W**, esto es, $W[0], \ldots, W[N_k - 1]$, corresponden a partes de la clave rellenas por columnas y de arriba a abajo
- El resto se definen de forma recursiva utilizando:
  1. la función `SubBytes` actuando sobre palabras, a la que llamaremos `SubWord`
  2. desplazamientos cíclicos y
  3. la operación $\oplus$.
- La recurrencia depende de la longitud de la clave.

Formación de las claves de ronda: Funciones

`SubWord` consiste en aplicar la S-box `SubBytes` a cada uno de los cuatro bytes de una palabra.

`RotWord` aplicado a una palabra de 4 bytes, devuelve una palabra cuyos bytes se han desplazado cíclicamente una posición a la izquierda, por ejemplo $\texttt{RotWord}(a, b, c, d) = (b, c, d, a)$. A veces también se llama `RotByte`.

`Rcon` $\texttt{Rcon}[i] = (RC[i], \texttt{0x00}, \texttt{0x00}, \texttt{0x00})$ siendo $RC[i]$ un elemento de $\mathbb{F}_{256}$ definido recursivamente mediante:

$$RC[1] = \texttt{0x01},$$
$$RC[i] = \texttt{0x02} \bullet RC[i-1],$$

que escrito como clases de polinomios es:

$$RC[1] = 1,$$
$$RC[i] = x \bullet RC[i-1]$$

Formación de las claves de ronda: $N_k = 4$ o 6

```
KeyExpansion(byte Key[4*Nk], word W[Nb*(Nr+1)])

   for(i=0; i<Nk; i++)
      W[i]=(Key[4*i],Key[4*i+1],Key[4*i+2],Key[4*i+3]);
   end for
   for(i=Nk; i<Nb*(Nr + 1); i++)
      temp=W[i-1];
      if (i mod Nk == 0)
         temp=SubWord(RotWord(temp)) XOR Rcon[i/Nk];
      W[i]=W[i-Nk] XOR temp;
   endfor
```

Formación de las claves de ronda: $N_k = 8$

```
KeyExpansion(byte Key[4*Nk], word W[Nb*(Nr+1)])

   for(i=0; i<Nk; i++)
      W[i]=(key[4*i],key[4*i+1],key[4*i+2],key[4*i+3]);
   end for
   for(i=Nk; i<Nb*(Nr+1); i++)
      temp=W[i - 1];
      if (i mod Nk == 0)
         temp=SubWord(RotWord(temp)) XOR  Rcon[i/Nk];
      else if (i mod Nk == 4)
         temp=SubWord(temp);
      end if
      W[i]=W[i-Nk] XOR temp;
   end for
```

La diferencia con la situación anterior es que cuando $i-4$ es un múltiplo de N_k, entonces aplicamos SubWord a $W[i-1]$ antes de hacer XOR.

Descifrado

Falta ver como se realiza el descifrado. Para esto se realizan las operaciones inversas de las realizadas y en orden inverso al realizado, es decir:
La primera ronda, ronda N_r, consiste en $\texttt{Round}(state, RoundKey_{N_r})$

1. $\texttt{InvAddRoundKey}(state, RoundKey_{N_r})$
2. $\texttt{InvShiftRows}(state)$
3. $\texttt{InvSubBytes}(state)$

A continuación se realizan $N_r - 1$ rondas con el siguiente esquema: $\texttt{Round}(state, RoundKey_i)$, $i = N_r - 1, \ldots, 1$

1. $\texttt{InvAddRoundKey}(state, RoundKey_i)$
2. $\texttt{InvMixColumns}(state)$
3. $\texttt{InvShiftRows}(state)$
4. $\texttt{InvSubBytes}(state)$

Y finalmente la última ronda con el esquema:

1. $\texttt{InvAddRoundKey}(state, RoundKey_0)$

Funciones inversas

`InvAddRoundKey` esta función es la misma que `AddRoundKey`, ya que el XOR es su propio inverso

`InvShiftRows` El mismo desplazamiento que `ShiftRows` pero a la derecha.

`InvSubBytes` primero se realiza la inversa de la transformación afín y después se calculan los inversos en $\mathbb{F}_{256}$.

`InvMixColumns` multiplicar por la matriz inversa de la usada en `MixColumn`:

$$\begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix}$$

Optimizaciones

Dos aspectos a considerar:

1. Las transformaciones `SubBytes` y `ShiftRows` (y en consecuencia sus inversas) conmutan, es decir, podemos cambiarlas de orden sin que afecte al resultado final
2. La multiplicación matricial es lineal, es decir,

$$\texttt{InvMixColumns}(state \oplus RoundKey_i) = \texttt{InvMixColumns}(state) \oplus \texttt{InvMixColumns}(RoundKey_i)$$

Definimos por este motivo las siguientes nuevas claves de ronda:

$$\begin{aligned} InvRoundKey_0 &= RoundKey_{N_r}, \\ InvRoundKey_i &= \texttt{InvMixColumn}(RoundKey_{N_r - i}) \text{ cuando } i = 1, \ldots, N_r - 1, \\ InvRoundkey_{N_r} &= RoundKey_0. \end{aligned}$$

Descifrado optimizado

El proceso de descifrado queda por tanto con la siguiente estructura
La ronda 0 consiste en

1. `AddRoundKey`$(state, InvRoundKey_0)$

A continuación se realizan $N_r - 1$ rondas con el siguiente esquema:
`Round`$(state, InvRoundKey_i)$, $i = 1, \ldots, N_r - 1$

1. `InvSubBytes`$(state)$
2. `InvShiftRows`$(state)$
3. `InvMixColumns`$(state)$
4. `AddRoundKey`$(state, InvRoundKey_i)$

Y finalmente la última ronda con el esquema:
`FinalRound`$(state, InvRoundKey_{N_r})$

1. `InvSubBytes`$(state)$
2. `InvShiftRows`$(state)$
3. `AddRoundKey`$(state, InvRoundKey_{N_r})$

Índice

One-time pad

Ya hemos hablado del *one-time pad*. Para mensajes digitales, es decir cadenas en el alfabeto $\mathbb{B} = \mathbb{Z}_2 = \mathbb{F}_2 = \{0, 1\}$, el sistema es conocido como cifrado de Vernam, ya que fue patentado por G. Vernam en 1919. Es el único criptosistema seguro de acuerdo con los criterios de Shannon. La clave es una secuencia aleatoria en $\mathbb{B}$ de longitud mayor que el mensaje, y la función de cifrado es la suma (XOR) del mensaje y la clave bit a bit.
Los cifrados de flujo emplean sucesiones pseudoaleatorias, conocidas aquí por el nombre de *sucesiones criptográficas*, en lugar de aleatorias. Estas sucesiones son generadas a partir de una *semilla*, tienen periodos muy grandes y es computacionalmente imposible averiguar un bit a partir de los anteriores, siempre que nos mantengamos por debajo de periodo. Las sucesiones pseudoaleatorias deben pasar algunos test estadísticos de aleatoriedad.

Postulados de Golomb I

Definición

- Sea $s = s_0 s_1 s_2 \ldots$ una sucesión infinita en $\mathbb{F}_2$. La cadena consistente en los primeros n términos de s se denota por $s^n = s_0 s_1 \ldots s_{n-1}$.
- Una sucesión s se dice N-periódica si $s_i = s_{i+N}$ para cualquier $i \geq 0$. Si s es N-periódica, el *ciclo* de s es s^N.
- Una racha de s es una subcadena de s consistente en 0s o 1s consecutivos delimitada por el otro símbolo. Una racha de 0s se llama hueco, y una racha de 1s bloque.
- Sea s una sucesión N-periódica. La función de autocorrelación de s es

$$C(t) = \sum_{i=0}^{N-1} (-1)^{s_{i+t}+s_i}, \text{ for } 0 \leq t \leq N-1.$$

Postulados de Golomb II

Definición

Sea s una sucesión N-periódica. Los postulados de Golomb son:

1. En el ciclo s^N, el número de 0s difiere del número de 1s en una unidad como máximo.
2. En el ciclo s^N, al menos la mitad de las rachas tienen longitud 1, al menos un cuarto tienen longitud 2, un octavo longitud 3, etc. Además, para cada longitud el número de huecos es (casi) igual al número de bloques.
3. La función de autocorrelación $C(t)$ fuera de fase es constante, i.e. existe $K \in \mathbb{Z}$ tal que para cualquier $0 < t \leq N - 1$

$$C(t) = K.$$

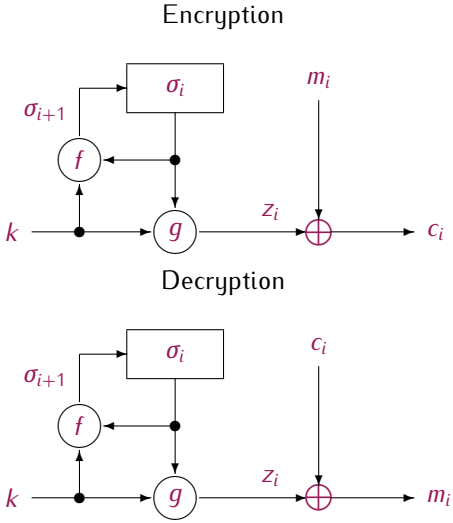
Cifrados de flujo síncronos

La sucesión criptográfica se genera independientemente del mensaje y del criptograma. Satisface las ecuaciones:

$$\begin{aligned} \sigma_{i+1} &= f(\sigma_i, k), \\ z_i &= g(\sigma_i, k), \\ c_i &= z_i \oplus m_i, \end{aligned}$$

donde σ_0 es el estado inicial, k es la clave, f es la función *siguiente estado*, g produce la sucesión criptográfica z_i que es sumada (XOR) con mensaje m_i para generar el criptograma c_i.

Cifrados de flujo síncronos: representación gráfica



Cifrados de flujo síncronos: propiedades

- Sincronización obligatoria.
- La inserción o borrado de bits se detecta fácilmente.
- La alteración de bits es difícil de detectar.

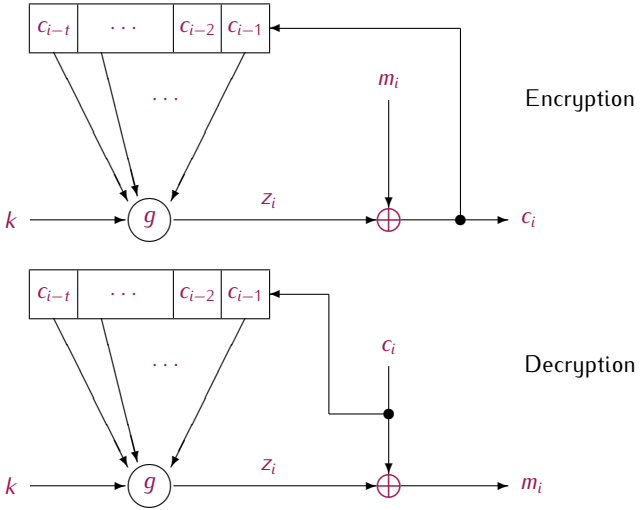
Cifrados de flujo autosincronizables

La sucesión criptográfica se genera a partir de la clave y de una cantidad fija de bits en el criptograma, matemáticamente

$$\begin{aligned}\sigma_i &= (c_{i-t}, c_{i-t+1}, \ldots, c_{i-1}),\\ z_i &= g(\sigma_i, k)\\ c_i &= z_i \oplus m_i,\end{aligned}$$

donde $\sigma_0 = (c_{-t}, c_{-t+1}, \ldots, c_{-1})$ es el estado inicial, k es la clave, g es la función que genera la sucesión criptográfica z_i que es sumada (XOR) con mensaje m_i para generar el criptograma c_i.

Cifrados de flujo autosincronizables: representación gráfica



Cifrados de flujo autosincronizables: características

- Sincronización automática.
- Inserción o borrado de bits más difícil de detectar.
- Alteración de bits más fácil de detectar.

Índice

Feedback shift registers

Registros lineales de desplazamiento retroalimentados (Linear feedback shift registers, LFSR)) son empleados usualmente para definir las funciones f y g. Algunas razones son las siguientes:

1. se implementan muy fácilmente en hardware,
2. producen sucesiones de periodo largo,
3. producen sucesiones con buenas propiedades estadísticas,
4. pueden ser analizados mediante técnicas algebraicas.

LFSR: estructura

Un LFSR se modela a partir de un polinomio $C(D) = 1 + c_1 D + c_2 D^2 + \cdots + c_l D^l \in \mathbb{F}_2[D]$. La sucesión se genera recursivamente

$$s_j = (c_1 s_{j-1} + c_2 s_{j-2} + \cdots + c_l s_{j-l}) \pmod 2 \quad \text{for } j \geq l$$

donde el estado inicial es $[s_0, s_1, \ldots, s_{l-1}]$.
Si $C(D)$ es un polinomio primitivo[1], cada uno de los $2^l - 1$ estados iniciales no nulos genera una sucesión del máximo periodo $2^l - 1$. Estas sucesiones satisfacen los postulados de Golomb y su complejidad lineal[2] es también alta.

[1] el polinomio mínimo de un elemento primitivo en la extensión de Galois $\mathbb{F}_2 \subseteq \mathbb{F}_{2^l}$.
[2] Un índice asociado a una sucesión y calculado por el algoritmo de Berlekamp-Masey.

LFSR: Problemas y soluciones

Como las funciones generadas por un LFSR son lineales, son vulnerables a ataques de texto en claro conocido. Hay varias soluciones:

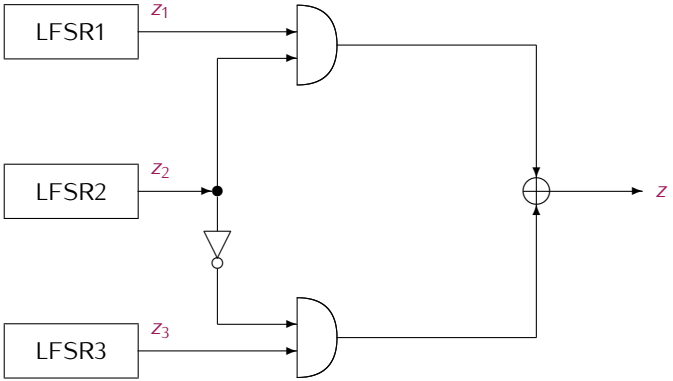
1. Podemos reemplazarlos por registros de desplazamiento retroalimentados no lineales (Non-linear feedback shift registers, NFSR),
2. podemos hacer una combinación no lineal de varios LFSRs,
3. podemos usar un filtro no lineal dentro de un LFSR,
4. un LFSR puede controlar el reloj que alimenta a varios LFSRs.

Generador de Geffe

Se utilizan tres LFSRs de longitudes l_1, l_2 y l_3 primos relativos. Las salidas z_1, z_2 y z_3 se combinan mediante la función no lineal

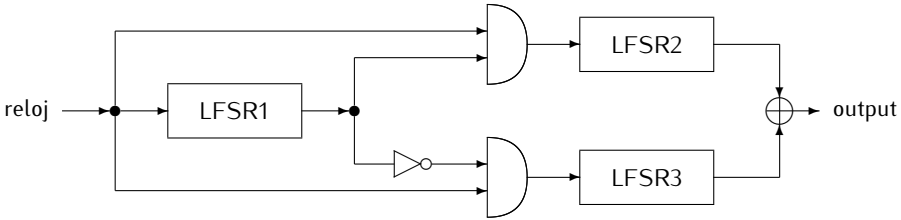
$$f(z_1, z_2, z_3) = z_1 z_2 \oplus \overline{z_2} z_3$$

El periodo global es $(2^{l_1} - 1)(2^{l_2} - 1)(2^{l_3} - 1)$, y la complejidad lineal $l_1 l_2 + l_2 l_3 + l_3$. Gráficamente



Generadores de paso alternado

Gráficamente:



Índice

eSTREAM

eSTREAM fue un proyecto, auspiciado por la red ECRYPT de la Unión Europea, desarrollado entre los años 2004 y 2008, y cuyo objetivo fue promover el diseño de cifrados de flujo compactos y eficientes adecuados para un uso generalizado. Como resultado se publicó en Abril de 2008, con una revisión en Septiembre, un portafolio que contiene siete cifrados de flujo adecuados para su uso en software o hardware.

Perfil 1 (SW)

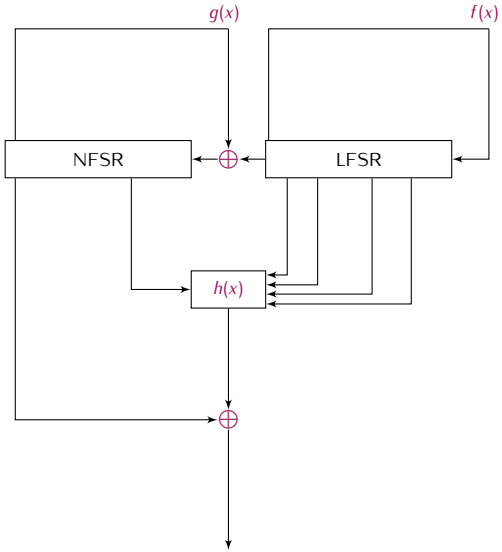
- HC-128
- Rabbit
- Salsa 20/12
- SOSEMANUK

Perfil 2 (HW)

- Grain v1
- MICKEY 2.0
- Trivium

`http://www.ecrypt.eu.org/stream/`

Grain v1: esquema gráfico



Grain v1: detalles

- $f(x) = 1 + x^{18} + x^{29} + x^{42} + x^{57} + x^{67} + x^{80}$
- $g(x) = 1 + x^{17} + x^{20} + x^{28} + x^{35} + x^{43} + x^{47} + x^{52} + x^{59} + x^{65} + x^{71} + x^{80} + x^{17}x^{20} + x^{43}x^{47} + x^{65}x^{71} + x^{20}x^{28}x^{35} + x^{47}x^{52}x^{59} + x^{17}x^{35}x^{52}x^{71} + x^{20}x^{28}x^{43}x^{47} + x^{17}x^{20}x^{59}x^{65} + x^{17}x^{20}x^{28}x^{35}x^{43} + x^{47}x^{52}x^{59}x^{65}x^{71} + x^{28}x^{35}x^{43}x^{47}x^{52}x^{59}$
- $h(x) = x_1 + x_4 + x_0x_3 + x_2x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4$
- $x_0, x_1, x_2, x_3, x_4 = s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}, b_{i+63}$

Resumen de características

Confidencialidad Es la principal característica, sólo la entidad autorizada tiene acceso a la información.

Autenticidad Sólo el poseedor de la clave puede haber cifrado la información.

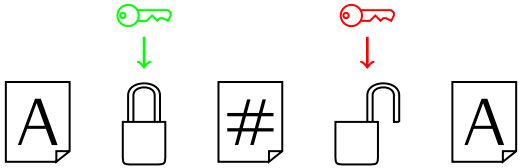
Integridad La alteración de la información se descubre al descifrar el criptograma, aunque algunos criptosistemas permiten reemplazar bloques completos de información.

No repudio No se puede garantizar, dos entidades comparten la clave.

Índice

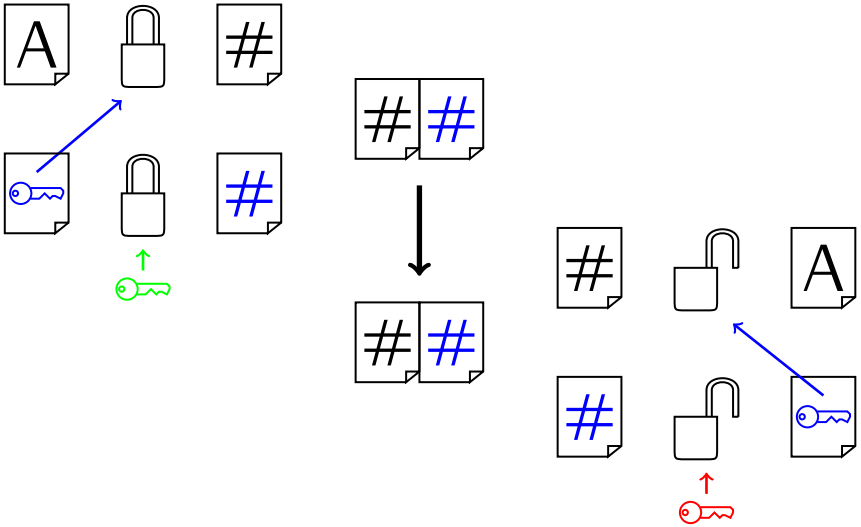
Criptosistemas asimétricos

Distinta clave cifra (pública) y descifra (privada).



1. Ventaja conceptual para Asimétricos: identificación e intercambio de claves.
2. Ventaja en coste para Simétricos: tamaño de las claves y velocidad.

Solución al coste: Criptosistemas híbridos



Análisis conceptual

Criptosistemas simétricos

- Canal seguro entre dos usuarios.
- Cada pareja de usuarios debe acordar una clave común.
- Permiten garantizar confidencialidad e integridad.

Criptosistemas asimétricos

- Cada usuario tiene una pareja de claves $(k, K) \in \mathcal{K}$, manteniendo K en privado y publicando k.
- Sus usos principales son distribución de claves, autenticidad y no repudio.

Construcción formal

- El espacio de claves es $\mathcal{K} = \mathcal{K}_p \times \mathcal{K}_s$.
- Existe una familia de aplicaciones $e_k : \mathcal{P} \to \mathcal{C}$, con $k \in \mathcal{K}_p$, para las que existe un algoritmo eficiente que las calcula pero es computacionalmente imposible calcular preimágenes.
- Estas funciones se llaman *funciones unidireccionales* o *one-way functions*.
- Para cada $k \in \mathcal{K}_p$ existe un $K \in \mathcal{K}_s$, que debe mantenerse en secreto, y una aplicación eficientemente computable $d_K : \mathcal{C} \to \mathcal{P}$, tal que $d_K(e_k(m)) = m$. K se llama *información trampa*.
- Las funciones unidireccionales con información trampa se llaman *funciones trampa* o *trapdoor function*.

Índice

One-way function: Potencias, raíces y logaritmos

Sean $n, a, e \in \mathbb{Z}$ positivos con $n \neq 0, 1$. Calcular a^e mód n es computacionalmente rápido mediante los cuadrados iterados, es decir, si $e = e_t e_{t-1} \dots e_1 e_0)_2$,

$$a^e = ((\cdots((a^{e_t})^2 a^{e_{t-1}})^2 a^{e_{t-2}} \cdots)^2 a^{e_1})^2 a^{e_0},$$

luego tenemos que hacer $2 \log_2 e + 1$ multiplicaciones para calcular potencias.

Sin embargo, dados $n, a, e \in \mathbb{Z}$, no hay en general un buen algoritmo para calcular $\sqrt[e]{a}$ mód n o $(\log_a e)$ mód n. Sí podemos calcular una raíz e-ésima de a si conocemos $\varphi(n)$, donde φ es la función indicatriz de Euler, y e es primo relativo con $\varphi(n)$. Concretamente, sea $d \in \mathbb{Z}$ tal que $ed \equiv 1$ mód $\varphi(n)$. El Teorema de Euler establece que $a^{\varphi(n)} \equiv 1$ mód n, de donde deducimos que $(a^d)^e = a^{ed} \equiv a$ mód n. Por tanto

$$a^d = \sqrt[e]{a} \text{ mód } n.$$

Descripción de RSA

El algoritmo de cifrado asimétrico RSA, acrónimo de Rivest, Shamir y Adleman, fue publicado en 1978. Se basa en la rapidez del cálculo de potencias y la lentitud del cálculo de raíces a no ser que conozcamos la función φ.

- Elegimos dos primos p, q suficientemente grandes. Calculamos $n = pq$.
- Calculamos $\varphi(n) = (p-1)(q-1)$.
- Elegimos $3 \leq e \leq \varphi(n)$ tal que $(e, \varphi(n)) = 1$.
- Calculamos $d = e^{-1} \pmod{\varphi(n)}$.
- La clave pública es el par (n, e).
- Mantenemos en privado p, q, d.
- La función de cifrado es

$$\mathrm{RSA}_{n,e}(m) = m^e \pmod{n}.$$

- La función de descifrado es

$$\mathrm{RSA}_{n,e}^{-1}(c) = \mathrm{RSA}_{n,d}(c) = c^d \pmod{n}.$$

Acelerando el cifrado y el descifrado

Cifrado

El método de cuadrados iterados es especialmente rápido si en la representación binaria del exponente hay muchos ceros. Una forma de lograr esto es elegir como e un primo de dichas características como $e = 3, 17, 2^{16} + 1$.

Descifrado

El Teorema Chino del Resto establece un isomorfismo

$$\phi : \mathbb{Z}_n \to \mathbb{Z}_p \times \mathbb{Z}_q$$
$$x \bmod n \mapsto (x \bmod p, x \bmod q).$$

Usando este isomorfismo,

$$c^d \bmod n = \phi^{-1}(c^{d \bmod p-1} \bmod p, c^{d \bmod q-1} \bmod q),$$

lo que reduce el tamaño de los números a emplear.

Seguridad de RSA

- Se cree que invertir la función $\mathrm{RSA}_{(n,e)}$ es un problema intratable.
- Conocer p, q nos lleva a conocer $\varphi(n) = (p-1)(q-1)$ y por tanto a calcular fácilmente d, por lo que en este caso sí podemos invertir $\mathrm{RSA}_{(n,e)}$.
- Conocer $\varphi(n)$ nos permite, obviamente, calcular d. En este caso

$$p + q = n - \varphi(n) + 1 \qquad p - q = \sqrt{(p+q)^2 - 4n}.$$

  Por lo tanto es computacionalmente equivalente conocer p, q y conocer $\varphi(n)$.
- Existen algoritmos polinomiales que factorizan n a partir de n, e, d. De nuevo conocer d se convierte en computacionalmente equivalente a factorizar $n = pq$.
- Hay situaciones en las que la estructura de p y q facilita encontrarlos a partir de n. Estas situaciones se evitan usando los llamados *primos fuertes*. Un número primo p es fuerte si
  - $p-1$ tiene un factor primo grande, llamado r,
  - $p+1$ tiene un factor primo grande,
  - $r-1$ tiene un factor primo grande.

  Se conjetura que son infinitos y fáciles de construir.

Índice

Logaritmo discreto, conjetura de Diffie y Hellman

- La potencia es una one way function en relación con el logaritmo, es decir, calcular g^a mód n es computacionalmente rápido, pero calcular $\log_g b$ mód n no lo es en general.
- Si los factores primos de n son pequeños, sí podemos calcular el logaritmo rápido usando el Teorema Chino del Resto. Para su uso en criptografía lo más útil es utilizar $n = p$ un número primo grande.
- Si g tiene pocas potencias distintas, también es rápido calcular logaritmos, por lo que es conveniente que g sea un generador de $\mathbb{Z}_p^*$, es decir, que todo elemento de $\mathbb{Z}_p$ distinto de 0 sea potencia de g.

Conjetura de Diffie y Hellman

Calcular g^{ab} mód p a partir de g^a mód p y g^b mód p es computacionalmente equivalente a calcular $a = \log_g g^a$ mód p o $b = \log_g g^b$ mód p.

ElGamal: generación de claves

- Seleccionamos aleatoriamente un número primo $p = 2rq + 1$ donde q es también primo grande y r tiene factores pequeños.
- Seleccionamos aleatoriamente g generador de $\mathbb{Z}_p^*$. Para que este proceso sea eficiente necesitamos que r tenga factores pequeños.
- Elegimos aleatoriamente $2 \leq x \leq p - 2$ y calculamos $y = g^x \bmod p$.
- La clave privada es (p, g, x), y la clave pública (p, g, y).

ElGamal: cifrado y descifrado

Cifrado

- El mensaje es un elemento $m \in \mathbb{Z}_p$.
- Aleatoriamente seleccionamos $2 \leq k \leq p-2$.
- El criptograma es

$$(c_1, c_2) = (g^k \bmod p, y^k m \bmod p).$$

Descifrado

Observemos que

$$c_1^{p-1-x} c_2 \equiv (g^k)^{p-1-x} y^k m \equiv (g^k)^{p-1-x} (g^x)^k m \equiv (g^{p-1})^k g^{-kx+xk} m \equiv m \mod p,$$

luego el descifrado es $c_1^{p-1-x} c_2 \bmod p$.

Seguridad de ElGamal

- El algoritmo de cifrado no es determinista. El criptograma depende de m, la clave pública (p, g, y) y de k, que es aleatorio para cada cifrado. Cifrar el mismo mensaje con la misma clave proporcionará dos criptogramas distintos.
- El atacante conoce $y = g^x$ y $c_1 = g^k$ para tratar de encontrar $y^k = g^{xk}$ con el que calcular $m = g^{-xk} c_2$.
- Si la conjetura de Diffie y Hellman es cierta, el atacante debe calcular $x = \log_g y \bmod p$ o $k = \log_g c_1 \bmod p$, computacionalmente difícil.

Índice

Conjetura de Diffie y Hellman en grupos

En realidad, para diseñar un sistema basado en la conjetura de Diffie y Hellman sólo hace falta una estructura multiplicativa y un elemento de orden finito, es decir, un grupo G y un elemento $g \in G$ tal que $g^n = 1$ para cierto n suficientemente grande.

Conjetura de Diffie y Hellman en grupos

Sea $g \in G$ un elemento de orden finito. Calcular g^{ab} a partir de g^a y g^b es computacionalmente equivalente a calcular $a = \log_g g^a$ o $b = \log_g g^b$.

La conjetura estándar es para $G = \mathbb{Z}_p^*$, con p un primo suficientemente grande.

Curvas elípticas en característica positiva

Característica impar

Una curva elíptica es el conjunto de puntos

$$E(\mathbb{Z}_p) = \left\{(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p \mid y^2 = x^3 + \alpha x + \beta\right\}$$

donde $4\alpha^3 + 27\beta^2 \neq 0$, junto con un punto $\mathcal{O}$ llamado punto del infinito. Esta es la forma de Weierstrass.

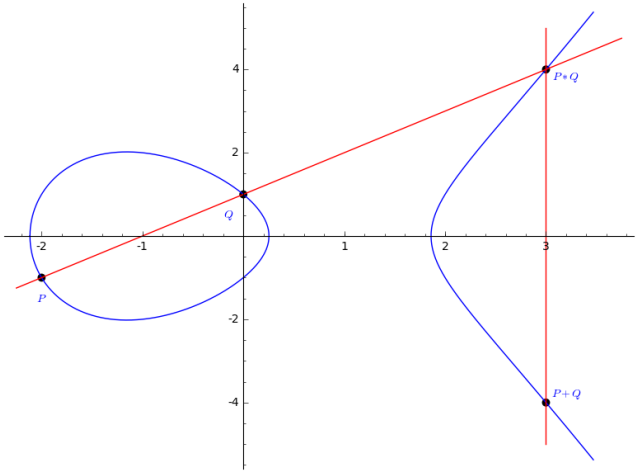
Característica 2

Una curva elíptica es el conjunto de puntos

$$E(\mathbb{F}_{2^l}) = \left\{(x, y) \in \mathbb{F}_{2^l} \times \mathbb{F}_{2^l} \mid y^2 + xy = x^3 + \alpha x^2 + \beta\right\}$$

donde $\beta \neq 0$, junto con un punto $\mathcal{O}$ llamado punto del infinito. Esta es una de las formas de Weierstrass en característica 2.

Aritmética en una curva elíptica I



Aritmética en una curva elíptica II

Aritmética en característica impar

Sean

$$E = \left\{(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p \mid y^2 = x^3 + \alpha x + \beta\right\},$$

$P = (x_0, y_0)$, $P_1 = (x_1, y_1)$ y $P_2 = (x_2, y_2)$.

- $-P = (x_0, -y_0)$.
- Si $P_2 = -P_1$, $P_1 + P_2 = \mathcal{O}$.
- Si $P_2 \neq -P_1$, $P_1 + P_2 = P_3$, viene dado por

$$P_3 = (x_3, y_3) = (\lambda^2 - x_1 - x_2, \lambda(x_1 - x_3) - y_1),$$

donde $\lambda = \dfrac{y_2 - y_1}{x_2 - x_1}$ si $P_1 \neq P_2$, y $\lambda = \dfrac{3x_1^2 + \alpha}{2y_1}$ si $P_1 = P_2$.

Aritmética en una curva elíptica III

Aritmética en característica 2

Sean

$$E = \left\{(x, y) \in \mathbb{F}_{2^l} \times \mathbb{F}_{2^l} \mid y^2 + xy = x^3 + \alpha x^2 + \beta\right\},$$

$P = (x_0, y_0)$, $P_1 = (x_1, y_1)$ y $P_2 = (x_2, y_2)$.

- $-P = (x_0, x_0 + y_0)$.
- Si $P_2 = -P_1$, $P_1 + P_2 = \mathcal{O}$.
- Si $P_2 \neq -P_1$, $P_1 + P_2 = P_3$, viene dado por

$$P_3 = (x_3, y_3) = (\lambda^2 + \lambda + \alpha + x_1 + x_2, \lambda(x_1 + x_3) + x_3 + y_1),$$

donde $\lambda = \dfrac{y_2 + y_1}{x_2 + x_1}$ si $x_1 \neq x_2$, y $\lambda = x_1 + \dfrac{y_1}{x_1}$ si $x_1 = x_2$.

Selección de curva y punto base

La curva se selecciona estableciendo los llamados parámetros de dominio:

- El cuerpo finito $\mathbb{F}_q$,
- los parámetros de la curva $\alpha, \beta \in \mathbb{F}_q$,
- un punto base $Q \in E(\mathbb{F}_q)$,
- el orden n de Q, es decir, el número $n > 0$ tal que $nQ = \mathcal{O}$ y $mQ \neq \mathcal{O}$ para cualquier $0 < m < n$,
- el cofactor h tal que $hn = |E(\mathbb{F}_q)|$.

La séxtupla $(\mathbb{F}_q, \alpha, \beta, Q, n, h)$ es pública.

Es recomendable que n sea un primo grande y que h sea pequeño. Hay procedimientos para lograr curvas variadas con todos estos requerimientos.

Conjetura de Diffie y Hellman en curvas elípticas

Dados unos parámetros $(\mathbb{F}_q, \alpha, \beta, Q, n, h)$, calcular $(ab)Q$ a partir de aQ y bQ es computacionalmente equivalente a calcular $a = \log_Q aQ$ o $b = \log_Q bQ$.

ElGamal en curvas elípticas

Generación de claves

- Fijamos unos parámetros $(\mathbb{F}_q, \alpha, \beta, Q, n, h)$.
- Elegimos aleatoriamente $1 \leq x \leq n-1$ y calculamos $P = xQ$.
- La clave privada es $(\mathbb{F}_q, \alpha, \beta, Q, n, h, x)$, y la clave pública $(\mathbb{F}_q, \alpha, \beta, Q, n, h, P)$.

Cifrado

- El mensaje es un elemento $m \in E(\mathbb{F}_q)$. Cómo realizar esta "codificación" no es evidente.
- Aleatoriamente seleccionamos $1 \leq k \leq n-1$.
- El criptograma es $(C_1, C_2) = (kQ, m + kP)$.

Descifrado

El descifrado es $C_2 - xC_1$ ya que

$$C_2 - xC_1 = m + kP - x(kQ) = m + k(xQ) - (xk)Q = m + (kx)Q - (kx)Q = m.$$

Curva P-192

- La curva se define en $\mathbb{F}_p$ donde

$$\begin{aligned} p &= 2^{192} - 2^{64} - 1 \\ &= 6277101735386680763835789423207666416083908700390324961279. \end{aligned}$$

- Tiene por ecuación

$$y^2 = x^3 - 3x + \beta,$$

donde

$$\beta = 0x\ 64210519\ e59c80e7\ 0fa7e9ab\ 72243049\ feb8deec\ c146b9b1.$$

- La curva tiene orden

$$6277101735386680763835789423176059013767194773182842284081.$$

Curva B-163

- La curva se define en $\mathbb{F}_{2^{163}}$ donde

$$\mathbb{F}_{2^{163}} = \mathbb{F}_2[x]_{x^{163}+x^7+x^6+x^3+1}$$

- Tiene por ecuación

$$y^2 + xy = x^3 + x^2 + \beta,$$

donde

$$\beta = 0x\ 00000002\ 0a601907\ b8c953ca\ 1481eb10\ 512f7874\ 4a3205fd.$$

- La curva tiene orden $2r$ donde

$$r = 5846006549323611672814742442876390689256843201587.$$

Resumen de características

Confidencialidad Es el objeto fundamental de los criptosistemas, garantizar confidencialidad, sólo emisor y receptor tienen acceso a la información.

Autenticidad Los cifrados de clave pública no garantizan autenticidad.

Integridad La alteración de la información se detecta.

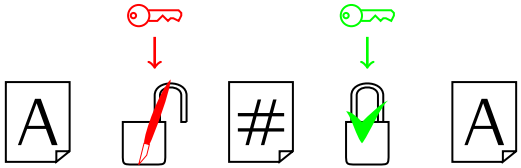
No repudio Tampoco se garantiza.

Índice

Autenticidad

Los criptosistemas de clave pública no garantizan la *autenticidad* del mensaje, no hay forma de saber si el mensaje o el remitente son auténticos. Tampoco puede garantizarse el *no repudio*, no hay forma de certificar que alguien ha enviado un mensaje.

La confidencialidad viene garantizada por el conocimiento de la clave privada. ¿Podemos utilizar este conocimiento para garantizar autenticidad? Si en un sistema RSA "cifamos" con la clave privada y "desciframos" con la clave pública, perdemos la confidencialidad pero obtenemos autenticidad y no repudio.



Índice

Concepto

Un sistema de firma es una quíntupla $(\mathcal{M}, \mathcal{F}, \mathcal{K}, \text{sgn}, \text{vfy})$ donde

- $\mathcal{M}$ es el espacio de los mensajes.
- $\mathcal{F}$ el conjunto de las firmas.
- $\mathcal{K}$ es el espacio de claves.
- sgn es la función firma $\text{sgn} : \mathcal{K} \times \mathcal{M} \to \mathcal{F}$.
- vfy es la función de verificación $\text{vfy} : \mathcal{K} \times \mathcal{M} \times \mathcal{F} \to \mathbb{Z}_2$
- Si $y = \text{sgn}_k(x)$ entonces $\text{vfy}_k(x, y) = 1$.
- Dados $k \in \mathcal{K}$ e $y \in \mathcal{F}$, es computacionalmente difícil calcular $x \in \mathcal{M}$ tal que $\text{vfy}_k(x, y) = 1$.

Para un sistema RSA,

$$\text{sgn}_{(n,d)}(m) = m^d \bmod n, \quad \text{vfy}_{(n,e)}(m, y) = \begin{cases} 1 & \text{si } y^e \equiv m \mod n, \\ 0 & \text{en otro caso.} \end{cases}$$

Índice

Necesidad

Con el esquema anterior la firma incrementa considerablemente el tamaño de la información. Dependiendo del criptosistema empleado, podemos duplicar el tamaño del mensaje, caso RSA, incluso triplicar, usando un esquema derivado de ElGamal. Necesitamos una herramienta que resuma un mensaje de longitud arbitraria a una cadena de tamaño fijo.

Las funciones hash criptográficas son la herramienta que necesitamos. Estas funcione se emplean en los siguientes escenarios:

- Protocolos de firma digital.
- Comprobación de la integridad de claves públicas.
- Generadores pseudoaleatorios.
- Usados con claves secretas, se convierten en códigos de autenticación de mensajes (MAC message authetication code).

Concepto

Una función hash es una función

$$h : \mathbb{B}^* \to \mathbb{B}^N$$

con las siguientes propiedades:

- Computacionalmente fácil de calcular, tanto en hardware como en software.
- Son funciones unidireccionales (*one-way*), es decir, dado $y \in \mathbb{B}^N$ es computacionalmente imposible encontrar $m \in \mathbb{B}^*$ tal que $h(m) = y$.
- Son funciones *resistentes a segundas preimágenes*, es decir, dado $m \in \mathbb{B}^*$ es computacionalmente imposible encontrar $m' \in \mathbb{B}^*$, $m' \neq m$ tal que $h(m) = h(m')$.
- Son *resistentes a colisiones*, es decir, es computacionalmente imposible encontrar $m, m' \in \mathbb{B}^*$ talen que $h(m) = h(m')$.

One-way y colisiones.

Proposición

Una función hash resistente a colisiones es unidireccional

Demostración.

Sea $n > N$ y consideremos $h : \mathbb{B}^n \to \mathbb{B}^N$. Supongamos que h no es unidireccional. Tenemos un algoritmo A que dado $y \in \mathbb{B}^N$ calcula $m \in \mathbb{B}^n$ con $h(m) = y$. Cojamos un $m \in \mathbb{B}^n$ aleatorio, y apliquemos A a $h(m)$ para calcular $m' \in \mathbb{B}^n$ tal que $h(m) = h(m')$. Calculemos la probabilidad de que $m \neq m'$. Sea $[m] = \{x \in \mathbb{B}^n \mid h(m) = h(x)\}$. La probabilidad de que $m' \neq m$ es $\frac{|[m]|-1}{|[m]|}$. La media de todas estas probabilidades será la probabilidad de encontrar una colisión. Esta media es:

$$\begin{aligned}\frac{1}{|\mathbb{B}^n|}\sum_{m\in\mathbb{B}^n}\frac{|[m]|-1}{|[m]|} &= \frac{1}{|\mathbb{B}^n|}\sum_{y\in\mathbb{B}^N}\sum_{m\in h^{-1}(y)}\frac{|[m]|-1}{|[m]|} = \frac{1}{|\mathbb{B}^n|}\sum_{y\in\mathbb{B}^N}\sum_{m\in h^{-1}(y)}\frac{|h^{-1}(y)|-1}{|h^{-1}(y)|}\\ &= \frac{1}{|\mathbb{B}^n|}\sum_{y\in\mathbb{B}^N}(|h^{-1}(y)|-1) = \frac{1}{|\mathbb{B}^n|}\sum_{y\in\mathbb{B}^N}|h^{-1}(y)| - \frac{1}{|\mathbb{B}^n|}\sum_{y\in\mathbb{B}^N}1\\ &= 1 - \frac{|\mathbb{B}^N|}{|\mathbb{B}^n|} = 1 - \frac{2^N}{2^n} = \frac{2^{n-N}-1}{2^{n-N}} \geq \frac{1}{2}.\end{aligned}$$

□

Paradoja del cumpleaños I

Sea $f : X \to Y$ una aplicación, supongamos $|X| = m$, $|Y| = M$. ¿Cuál es la probabilidad de, elegidos t elementos en X, al menos dos de ellos tengan la misma imagen?

Calculamos el suceso contrario. La probabilidad de que las imágenes de t elementos de X sean distintas es:

$$\left(1 - \tfrac{1}{M}\right)\left(1 - \tfrac{2}{M}\right)\cdots\left(1 - \tfrac{t-1}{M}\right) = \prod_{i=1}^{t-1}\left(1 - \tfrac{i}{M}\right) \approx \prod_{i=1}^{t-1} e^{-\frac{i}{N}} = e^{-\frac{t(t-1)}{2M}}.$$

La probabilidad buscada es

$$1 - e^{-\frac{t(t-1)}{2M}} = \varepsilon.$$

Despejando de esta expresión el valor de t se obtiene:

$$t(t-1) = 2M \ln(\tfrac{1}{1-\varepsilon}),$$

de donde:

$$t \approx \sqrt{2M \ln(\tfrac{1}{1-\varepsilon})}.$$

Paradoja del cumpleaños II

Obtenemos que el valor de t es directamente proporcional a la raíz cuadrada de M.
Para $\varepsilon = \frac{1}{2}$ y $M = 365$,

$$t \approx \sqrt{2 \cdot 365 \ln(2)} = \sqrt{\ln(4)}\sqrt{365} = 1,17 \cdot 19,105 = 22,4944.$$

Para $\varepsilon = \frac{1}{2}$ y $M = 2^N$,

$$t \approx \sqrt{2 \cdot 2^N \ln(2)} \approx 1,177 \cdot 2^{\frac{N}{2}},$$

por tanto

valor de 2^N	valor de t
2^{20}	$2^{10} \approx 10^3$
2^{40}	$2^{20} \approx 10^6$
2^{80}	$2^{40} \approx 10^{12}$
2^{160}	$2^{80} \approx 10^{24}$

Ejemplo basado en el logaritmo discreto I

- q primo tal que $p = 2q + 1$ también es primo.
- $g_1, g_2 \in \mathbb{Z}_p$ elementos primitivos.
- Definimos

$$h : \mathbb{Z}_q \times \mathbb{Z}_q \longrightarrow \mathbb{Z}_p \setminus \{0\}, \quad h(x_1, x_2) = g_1^{x_1} g_2^{x_2} \text{ mód } p.$$

- Si $h(x_1, x_2) = h(y_1, y_2)$,

$$g_1^{x_1} g_2^{x_2} \equiv g_1^{y_1} g_2^{y_2} \quad \text{mód } p,$$

de donde

$$g_1^{x_1 - y_1} \equiv g_2^{y_2 - x_2} \quad \text{mód } p.$$

Llamemos $\log_{g_1}(g_2) = l$, es decir, $g_1^l \equiv g_2 \mod p$. Encontrar l es computacionalmente difícil. Como

$$g_1^{x_1 - y_1} \equiv g_1^{l(y_2 - x_2)} \quad \text{mód } p,$$

Ejemplo basado en el logaritmo discreto II

l es una solución de una de las ecuaciones

$$x_1 - y_1 \equiv l(y_2 - x_2) \mod 2q$$
$$x_1 - y_1 \equiv l(y_2 - x_2 + q) \mod 2q.$$

Encontrar una colisión es equivalente a resolver el logaritmo discreto.

Ejemplo

La construcción anterior para $q = 509$, $p = 1019$, $g_1 = 321$ y $g_2 = 792$ presenta la siguiente colisión:

$$h(373, 136) = h(162, 107),$$

con lo que es posible calcular $\log_{g_1}(g_2)$.

Construcción de Merkle-Damgård

Esta construcción permite extender una función (llamada *función de compresión*) $f : \mathbb{B}^{N+r} \to \mathbb{B}^N$ resistente a colisiones a una función $h : \mathbb{B}^* \to \mathbb{B}^N$ resistente a colisiones. Sea $m \in \mathbb{B}^*$.

- Añadimos a m un 1. Añadimos después el menor número de 0 necesario para obtener $\tilde{m} = m||10\cdots0 \in \mathbb{B}^{kr}$.
- Descomponemos $\tilde{m} = m_1||m_2||\ldots||m_k$ con $m_i \in \mathbb{B}^r$ para cada $1 \leq i \leq k$.
- Añadimos un nuevo bloque $m_{k+1} \in \mathbb{B}^r$ que contiene el tamaño inicial de m, luego $\tilde{m} = m_1||\ldots||m_k||m_{k+1}$.[3]
- Fijamos un valor inicial $v_0 \in \mathbb{B}^N$, que puede ser fijo para todos los mensajes.
- Para cada $1 \leq i \leq k+1$, $v_i = f(v_{i-1}||m_i)$.
- Definimos $h(m) = v_{k+1}$.

Proposición

Si f es resistente a colisiones, h es resistente a colisiones.

Demostración.

Ver [DK15, Proposition 2.14]. □

[3] El último bloque evita encontrar colisiones en caso de que $v_0 = v_i$ para algún i

Construcción esponja

En este caso construimos $h : \mathbb{B}^* \to \mathbb{B}^N$ a partir de una función biyectiva $f : \mathbb{B}^r \times \mathbb{B}^c \to \mathbb{B}^r \times \mathbb{B}^c$. Llamemos $l = \left\lceil \frac{N}{r} \right\rceil$. Sea $m \in \mathbb{B}^*$.

- Transformamos m en $\tilde{m} = m_1||m_2||\dots||m_k$ como en la construcción de Merkle-Damgård.
- Reservamos un registro estado $s = (s_1, s_2) \in \mathbb{B}^r \times \mathbb{B}^c$, que inicialmente asignamos $s = (0 \cdots 0, 0 \cdots 0)$.
- Fase absorción. Para cada $1 \leq i \leq k$,

$$s \leftarrow f(s_1 \oplus m_i, s_2)$$

- Fase exprimido. Para cada $1 \leq i \leq l$,

$$g_i \leftarrow s_1, s \leftarrow f(s).$$

- La salida es $g_1||\dots||g_l$ truncada a N bits.

Funciones Hash reales I

En desuso

Basadas en la construcción de Merkle-Damgård, vienen siendo utilizadas desde los años 90. Destacamos las siguientes.

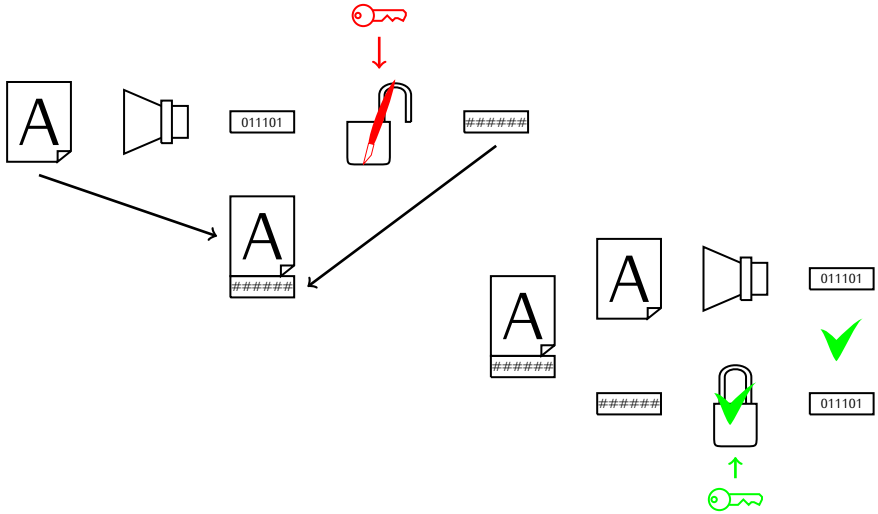
- MD4, diseñado por Ron Rivest en los 90, aunque totalmente desaconsejado por el pequeño tamaño de la salida. Sus principios de diseño se encuentran en algunas de las más populares como MD5, SHA-1 y RIPEMD-160.
- MD5 tiene una salida de 128 bits, SHA-1 y RIPEMD-160 tienen una salida de 160 bits. La tasa de compresión es $r = 512$ bits.
- Hasta Octubre de 2012 el estándar era la familia SHA publicada en [Nat02]. Esta familia incluye SHA-1, SHA-256, SHA-384 y SHA-512, los tres últimos agrupados en el nombre genérico SHA-2. Sus salidas son 160, 256, 384 y 512 bits respectivamente.
- Se han encontrado colisiones para MD5, SHA-1 y RIPEMD-160. SHA-2 sigue siendo resistente a colisiones. Hay muchos menos ataques a la unidireccionalidad.

Funciones Hash reales II

Nuevo estándar

- Concurso propuesto por el NIST en Noviembre de 2007 ante la rapidez con la que se iban produciendo avances en la búsqueda de colisiones en SHA.
- Los candidatos finales fueron BLAKE, Grøstl, JH, Keccak and Skein.
- El ganador ha sido Keccak, diseñado por Bertoni, Daemen, Peeters y Van Assche.
- Keccak, renombrado como SHA-3 (*Secure Hash Algorithm-3*), está basado en una construcción esponja.
- La permutación f actúa sobre un tamaño $b = r + c = 1600$ bits.
- Las posibles salidas son 224, 256, 384 y 512 bits.
- Las tasas de bits correspondientes son $r = 1152,\ 1088,\ 832,\ 576$ respectivamente en función de la salida.

Uso de funciones hash: Firma y verificación digital.



Códigos de detección de modificaciones (MDC)

- Las funciones hash también son conocidas con el nombre de funciones resumen del mensaje, message digest en inglés, y al valor $h(m)$ el resumen, huella dactilar, digest, fingerprint, thumbprint, etc.
- Las funciones hash criptográficas pueden usarse para garantizar la integridad de un mensaje, almacenando el valor $h(m)$ en un lugar seguro. La alteración de m puede detectarse calculando su hash y comparando con el valor almacenado.
- Cuando son usadas así, las funciones hash reciben el nombre de códigos de detección de modificaciones, modification detection codes (MDC) en inglés.
- Por ejemplo, claves públicas, certificados digitales, que veremos más adelante, o paquetes de software suelen llevar asociado una huella dactilar para verificar su integridad, huella publicada en una web razonablemente segura. Incluso puede solicitarse por correo ordinario en ocasiones.

Códigos de Autenticación de Mensajes (MAC)

MAC

La autenticación de un mensaje tiene como objetivo autentificar el origen del mismo y garantizar, al mismo tiempo, la integridad. Se emplean en protocolos como SSL/TLS e IPSec, que veremos más adelante. Un MAC depende de una clave secreta compartida por las dos entidades participantes en una comunicación por lo que las dos entidades pueden generar un MAC válido.

Hay dos técnicas estándar para fabricar MAC

- **HMAC** Basado en funciones hash criptográficas.
- **CBC-MAC** Basado en criptosistemas simétricos.

HMAC

Publicado en [Dan08], es un método estándar para producir un HMAC a partir de una función hash h, derivada de una función de compresión f mediante la construcción de Merkle-Damgård.

- La longitud de los hash y la tasa de compresión son múltiplos de 8, por lo que los medimos en bytes.
- La clave secreta k tiene longitud no mayor de r bytes, extendida a r bytes añadiendo ceros.
- Definimos dos constantes $ipad = 0\text{x}\underbrace{36\cdots36}_{r}$ y $opad = 0\text{x}\underbrace{5\text{C}\cdots5\text{C}}_{r}$
- El HMAC de un mensaje m es

$$\text{HMAC}(k, m) = h((k \oplus opad)||h((k \oplus ipad)||m))$$

- Aplicar la función hash dos veces previene contra un ataque conocido como ataque de extensión de longitud: Si $\text{HMAC}(k, m) = h((k \oplus ipad)||m))$, sea $v_0 = h((k \oplus ipad)||m))$, aplicamos la construcción Merkle-Damgård a un mensaje m', pero añadiendo el tamaño de $\tilde{m}||m'$ en lugar del tamaño de m'. La salida de esta construcción es $\text{HMAC}(k, m')$.

CBC-MAC

- El MAC de un mensaje m es el último criptograma obtenido al aplicar un criptosistema simétrico $e_k : \mathbb{B}^N \to \mathbb{B}^N$ en modo CBC y clave secreta k.
- El vector de inicialización es típicamente $IV = 0\text{x}0 \cdots 0$.
- Partimos m en l bloques de longitud N $m = m_1||m_2|| \cdots ||m_l$.
- Calculamos

$$\begin{aligned} c_0 &= IV \\ c_i &= e_k(m_i \oplus c_{i-1}),\ i = 1, \ldots, l \\ \text{CBC-MAC}(k, m) &= c_l \end{aligned}$$

Índice

Estándar

- El DSA (Digital Signature Algorithm) fue propuesto por el NIST en 1991, y adoptado como DSS (Digital Signature Standard) en 1993. Desde entonces se han propuesto cuatro revisiones del mismo, la última en 2013 [Nat13].
- Es muy similar al esquema de firma digital desarrollado a partir del criptosistema de ElGamal.

Generación de claves

- Se elige un primo q de N bits.
- Se elige un primo p de L bits tal que $p = 2kq + 1$. Las combinaciones N, L permitidas son

L	N
1024	160
2048	224
2048	256
3072	256

- Elegimos $h \in \mathbb{Z}_p^*$ tal que $h^{(p-1)/q} \not\equiv 1 \mod p$. Llamamos $g = h^{(p-1)/q} \mod p$.
- Elegimos $1 \leq x \leq q-1$ aleatoriamente. Calculamos $y = g^x \mod p$.
- La clave privada (usada para firmar) es (p, q, g, x).
- La clave pública (usada para verificar) es (p, q, g, y).

Proceso de firma

- El mensaje es $m \in \mathbb{Z}_q$. Se obtiene como el resultado de aplicar una función hash a un mensaje mayor.
- Seleccionamos aleatoriamente $1 \leq k \leq q$.
- Calculamos $r \leftarrow (g^k \bmod p) \bmod q$ y $s \leftarrow k^{-1}(m + rx) \bmod q$. Si $s = 0$ volvemos a seleccionar k, pero es muy improbable que ocurra.
- El mensaje firmado es (m, r, s).

Verificación

- El receptor conoce el mensaje firmado (m, r, s) y la clave pública (p, q, g, y).
- Calculamos $t \leftarrow s^{-1} \bmod q$ y $v \leftarrow ((g^m y^r)^t \bmod p) \bmod q$.
- La verificación es afirmativa si y sólo si $v = r$.

Proposición

Si la firma es correcta, $v = r$.

Demostración.

$$\begin{aligned} v &= ((g^m y^r)^t \bmod p) \bmod q \\ &= (g^{mt} g^{rxt} \bmod p) \bmod q \\ &= (g^{(m+rx)t \bmod q} \bmod p) \bmod q \\ &= (g^k \bmod p) \bmod q \\ &= r \end{aligned}$$

donde los exponentes se calculan módulo q. □

Índice

Diffie-Hellman I

Las dos entidades que desean compartir una clave son Alicia y Bernabé.

1. Alicia y Bernabé fijan un primo p tal que el problema del logaritmo discreto es intratable en $\mathbb{Z}_p^*$, y un elemento primitivo $g \in \mathbb{Z}_p$.
2. Alicia escoge aleatoriamente $1 \leq a \leq p-2$, calcula $c = g^a \bmod p$ y lo envía a Bernabé.
3. Bernabé escoge aleatoriamente $1 \leq b \leq p-2$, calcula $d = g^b \bmod p$ y lo envía a Alicia.
4. Alicia calcula la clave compartida $k = d^a = g^{ab} \bmod p$.
5. Bernabé calcula la clave compartida $k = c^b = g^{ab} \bmod p$.

La seguridad se basa en la conjetura de Diffie-Hellman.

Diffie-Hellman II

Formalmente, tenemos los siguientes elementos:

1. Un sistema de clave pública que genera aleatoriamente parejas de claves (K, k) privada/pública a partir de unos parámetros públicos p mediante una función $(K, k) = \text{gen}(p, \texttt{rand})$.
2. Una función $\text{Der}(K_1, k_2)$ que deriva una cadena a partir de una clave pública y otra privada.

El protocolo se describe por tanto de la siguiente forma

1. Alicia y Bernabé fijan unos parámetros p.
2. Alicia genera aleatoriamente $(K_A, k_A) = \text{gen}(p, \texttt{rand})$. Envía k_A a Bernabé
3. Bernabé genera aleatoriamente $(K_B, k_B) = \text{gen}(p, \texttt{rand})$. Envía k_B a Alicia.
4. Alicia calcula $k_c = \text{Der}(K_A, k_b)$.
5. Bernabé calcula $k'_c = \text{Der}(K_B, k_A)$.

El sistema funciona siempre que $k_c = k'_c$ y siempre que no sea posible calcular k_c a partir de k_A y k_B.

Este protocolo no proporciona autenticación de las partes, y es vulnerable a ataques activos del "hombre en el medio".

Estación a estación

Mejora del protocolo de Diffie-Hellman buscando resolver sus debilidades. Fijamos $g \in \mathbb{Z}_p^*$ como en el protocolo de Diffie-Hellman. Asumimos que tenemos disponible un esquema de firma digital. Cada usuario tiene una pareja de claves (s, v) privada-pública para firma y verificación. Las partes públicas son auténticas y accesibles.

1. Alicia escoge aleatoriamente $1 \leq a \leq p-2$, calcula $c = g^a \bmod p$ y lo envía a Bernabé.
2. Bernabé escoge aleatoriamente $1 \leq b \leq p-2$, calcula $d = g^b \bmod p$ y $k = g^{ab} = (c)^b \bmod p$. Calcula $s = \mathrm{sgn}_{s_B}(c||d)$. Envía $(d, e_k(s))$ a Alicia.
3. Alicia calcula $k = g^{ab} = d^a \bmod p$ y $s = d_k(e_k(s))$, verifica $\mathrm{vfy}_{v_b}(c||d, s)$. Firma $t = \mathrm{sgn}_{s_A}(d||c)$ y envía a Bernabé $e_k(t)$.
4. Bernabé calcula $t = d_k(e_k(t))$ y verifica $\mathrm{vfy}_{v_A}(d||c, t)$.
5. A partir de este momento ambos pueden estar seguros de que la clave secreta k está compartida sólo por ellos dos.

Kerberos (simplificado)

Desarrollado en el MIT, es un ejemplo de protocolo con árbitro, una entidad especial en la que todas las demás confían. En este protocolo el árbitro recibe el nombre de *Kerberos authentication server*, y lo denotamos T. Éste comparte una clave para un criptosistema simétrico e con cada cliente A, k_A, y cada servidor B, k_B.

1. A elige aleatoriamente r_A y envía a T la petición (A, B, r_A).
2. T genera una clave de sesión k y crea un ticket $t = (A, k, l)$, donde l define un periodo de validez para el ticket. Envía $(e_{k_A}(k, r_A, l, B), e_{k_B}(t))$ a A.
3. A recupera $(k, r_A, l, B) = d_{k_A}(e_{k_A}(k, r_A, l, B)))$. Verifica que B y r_A son los que envió. Crea un autentificador $a = (A, t_A)$ donde t_A es un sello de tiempo del reloj local de A. Envía $(e_k(a), e_{k_B}(t))$ a B.
4. B recupera $t = (A, k, l) = d_{k_B}(e_{k_B}(t))$ y $a = (A, t_A) = d_k(e_k(a))$. Comprueba que
   1. el identificador A es el mismo en el ticket t y en el autentificador a,
   2. el sello t_A es actual, en un pequeño entorno del reloj local de B,
   3. el sello t_A está dentro de los límites de l.

   Si todos estos pasos son verificados, B considera A autenticado.
5. B envía $e_k(t_A)$ a A.
6. A compara $d_k(e_k(t_A))$ con el valor de t_A del autentificador. Si coinciden considera B autenticado.

Índice

Comprobación interactiva de conocimiento

- Hay dos participantes en uno de estos protocolos, el sujeto P (*prover* en inglés) y el sujeto V (*verifier* en inglés). P tiene conocimiento de algún dato y quiere convencer a V de que conoce dicho dato. Alternativamente, P y V realizan una serie de *movimientos* al final de los cuales V acepta o rechaza la comprobación de P.
- Hay dos requerimientos para un sistema de comprobación interactiva:

  Completitud Si P conoce el dato, V siempre aceptará la comprobación de P.

  Robustez Si P puede convencer a V con una probabilidad razonable, conoce el dato.
- Los sujetos son llamados *honestos* si siguen el comportamiento diseñado en el protocolo. En caso contrario se llaman *tramposos*. La trampa no está en alterar la sintaxis de la comunicación, sino en modificar los mensajes transmitidos.
- El esquema más básico consiste en que P envíe el dato a V, pero eso obliga a que V también conozca el dato, y cualquiera que vigile la comunicación vería el dato. Este esquema es asimilable a enviar una contraseña.

Conocimiento de una clave privada

P debe probar ante V que conoce la clave privada K asociada a una clave pública conocida k. El protocolo consta de dos movimientos:

1. V selecciona un mensaje aleatorio m, calcula $c = e_k(m)$ y envía c a P.
2. P calcula $m' = d_K(c)$ y lo envía a V.
3. V acepta si y sólo si $m = m'$.

- Completitud y robustez son evidentes.
- Un adversario que observe la comunicación difícilmente podrá suplantar más adelante a P por la aleatoriedad de m.
- Si V es tramposo y envía a P en el primer paso un criptograma obtenido de una tercera parte, obtendrá el texto en claro de dicho criptograma sin necesidad de averiguar la clave.

Protocolo de Fiat-Shamir simplificado I

- P selecciona $n = pq$ con p, q primos aleatorios. La habilidad de calcular raíces cuadradas en $\mathbb{Z}_n$ es equivalente a la habilidad de factorizar n.
- P elige aleatoriamente $y \in \mathbb{Z}_n$ y calcula $x = y^2 \bmod n$, es decir y es la raíz cuadrada de x en $\mathbb{Z}_n$. La pareja (n, x) es pública, mientras que (p, q, y) se mantienen en secreto.
- P debe convencer a V de que conoce y, una raíz cuadrada de x.

Fiat-Shamir simplificado

1. P elige aleatoriamente $r \in \mathbb{Z}_n^*$, calcula $a = r^2 \bmod n$ y lo envía a V.
2. V elige aleatoriamente $e \in \{0, 1\}$ y lo envía a P.
3. P calcula $b = ry^e \bmod n$ y lo envía a V.
4. V acepta si $b^2 \equiv ax^e \mod n$.

Completitud Si P conoce y, ry^e es la raíz cuadrada de ax^e, luego V acepta.

Robustez Un tramposo E puede convencer a V de que conoce x con probabilidad $\frac{1}{2}$:

Protocolo de Fiat-Shamir simplificado II

1. E elige aleatoriamente $r \in \mathbb{Z}_n^*$ y $f \in \{0,1\}$, calcula $a = r^2 x^{-f} \mod n$ y lo envía a V.
2. V elige aleatoriamente $e \in \{0,1\}$ y lo envía a E.
3. E envía r a V.

V acepta si y sólo si $e = f$, lo que ocurre con probabilidad $\frac{1}{2}$.

- Un tramposo E tratando de suplantar a P no puede hacerlo con probabilidad mayor que $\frac{1}{2}$. De ser así, E conocería un valor de a para el cual puede contestar correctamente a ambos retos, es decir, conoce b_1 y b_2 tales que

$$b_1^2 \equiv a \mod n \text{ y } b_2^2 \equiv ax \mod n.$$

  En este caso puede calcular una raíz cuadrada de x, $y = b_2 b_1^{-1} \mod n$, lo que es intratable.
- Ni V ni E tienen algún conocimiento del valor de y.

Índice

Protocolo de conocimiento cero

Un protocolo de conocimiento cero (o mínimo) es aquel en el que cualquier cálculo que un verificador V, honesto o no, puede eficientemente realizar después de interactuar con P, puede ser eficientemente simulado sin interacción.

El protocolo de Fiat-Shamir simplificado es un ejemplo de protocolo de conocimiento cero.

Protocolo de Fiat-Shamir I

- P selecciona $n = pq$ con p, q primos aleatorios. La habilidad de calcular raíces cuadradas en $\mathbb{Z}_n$ es equivalente a la habilidad de factorizar n.
- P elige aleatoriamente $y = (y_1, \ldots, y_t) \in \mathbb{Z}_n^t$ y calcula $x = (y_1^2 \bmod n, \ldots, y_t^2 \bmod n)$. La lista (n, x) es pública, mientras que (p, q, y) se mantiene en secreto.
- P debe convencer a V de que conoce y.

Protocolo de Fiat-Shamir

Repetimos el siguiente proceso k veces:

1. P elige aleatoriamente $r \in \mathbb{Z}_n^*$, calcula $a = r^2 \bmod n$ y lo envía a V.
2. V elige aleatoriamente $e = (e_1, \ldots, e_t) \in \{0, 1\}^t$ y lo envía a P.
3. P calcula $b = r y_1^{e_1} \cdots y_t^{e_t} \bmod n$ y lo envía a V.
4. V rechaza si $b^2 \not\equiv a x_1^{e_1} \cdots x_t^{e_t} \bmod n$ y detiene el protocolo.

Completitud Si P conoce y, $r y_1^{e_1} \cdots y_t^{e_t}$ es la raíz cuadrada de $a x_1^{e_1} \cdots x_t^{e_t}$, luego V acepta.

Protocolo de Fiat-Shamir II

Robustez Para que un tramposo realice una suplantación similar a la del protocolo simplificado, debe realizar kt elecciones aleatorias en el conjunto $\{0,1\}$, por lo que la probabilidad debe ser 2^{-kt}.

- Un tramposo E tratando de suplantar a P no puede hacerlo con probabilidad mayor que 2^{-kt}. De ser así, E conocería k valores de $a^1,\dots,a^k$ para el cual puede contestar correctamente a dos retos retos diferentes $(e^1,\dots,e^k)$ y $(f^1,\dots,f^k)$. Hay una iteración j para la cual $e^j \neq f^j$, es decir, puede calcular b_1 y b_2 tales que

$$b_1^2 \equiv a x_1^{e_1^j} \dots x_t^{e_t^j} \mod n \text{ y } b_2^2 \equiv a x_1^{f_1^j} \dots x_t^{f_t^j} \mod n.$$

  En este caso $b_2 b_1^{-1}$ mód n es una raíz cuadrada del elemento aleatorio $x_1^{f_1^j - e_1^j} \cdots x_t^{f_t^j - e_t^j}$, lo que es intratable.
- Se puede demostrar que para $t \in \mathcal{O}(\log_2(|n|))$ y $k \in \mathcal{O}(|n|^l)$, el protocolo sigue siendo de conocimiento cero.

Índice

Esquemas de umbral

Un dato secreto se trocea en n piezas de manera segura y se reparte entre el mismo número de usuarios. Una coalición de algunos de los usuarios debe ser capaz de recuperar el dato secreto.

Esquema (t, n)-umbral.

Un centro de confianza trocea el secreto en n piezas que reparte entre n usuarios. Cualesquiera t usuarios juntos deben ser capaces de recuperar el valor del secreto, mientras que $t-1$ o menos usuarios juntos deben ser incapaces de recuperarlo.

Esquema umbral de Shamir

El centro de confianza T reparte un secreto $s \in \mathbb{Z}$ entre n usuarios $\{P_1, \ldots, P_n\}$.

1. T escoge un primo $p \geq \max\{n, s\}$ y asigna $a_0 \leftarrow s$.
2. T selecciona aleatoriamente $a_1, \ldots, a_{t-1} \in \mathbb{Z}_p$ y obtiene el polinomio $f(x) = \sum_{i=0}^{t-1} a_i x^i \in \mathbb{Z}_p[x]$.
3. T escoge $x_1, \ldots, x_n \in \mathbb{Z}_p$ distintos, para cada i calcula $s_i \leftarrow f(x_i)$ y envía la pareja (x_i, s_i) a P_i de manera segura.

- Como $\deg f(x) = t - 1$, son necesarios t puntos para calcularlo usando, por ejemplo, el polinomio de interpolación de Lagrange. Con menos de t valores no es posible saber el polinomio, y todos los valores $a \in \mathbb{Z}_p$ pueden aparecer con menos de t puntos con la misma probabilidad.
- Se puede extender con facilidad a nuevos usuarios.
- Un usuario puede tener uno o más trozos.

Índice

Índice

Claves públicas: propietarios y validez

Una clave pública:

- ¿Ha sido alterada?
- ¿Pertenece a quien dice ser su propietario?
- ¿Para qué funciones sirve?
- ¿Continúa siendo válida?

Los certificados digitales y las infraestructuras de clave pública (PKI) dan respuesta a estas y otras preguntas.

Qué es un certificado

Un certificado de llave pública es una afirmación, firmada digitalmente por una entidad emisora, que asegura que la clave pública (e información adicional) de la entidad propietaria del certificado tiene un valor específico.

Un certificado digital contiene:

- Claves públicas del propietario.
- Información del propietario (por ejemplo datos del usuario tales como el nombre, su identificador, etc.)
- Información de los emisores.
- Una o más firmas digitales de los emisores.

Si confiamos en la entidad que emite el certificado y éste es auténtico tenemos la garantía de que la información contenida en el certificado en veraz, en particular las claves públicas en él contenidas pertenecen a la entidad propietaria del mismo.

Índice

Anillos de confianza

Todos los participantes del anillo pueden convertirse en emisores de certificados. Es el esquema empleado en PGP y sus derivados como GnuPG y OpenPGP.

- Permiten establecer niveles de confianza y de extensión.
- Su fortaleza iguala a la menor de sus participantes.
- No necesitan infraestructura adicional.
- Cada certificado puede estar firmado por más de un emisor.
- Puede ser difícil encontrar la información necesaria para verificar una firma.

Anillos de confianza: transitividad

Uno de los elementos que los anillos de confianza pretenden destacar es la "transitividad" de las claves firmadas. La idea es que debería considerar como válida una clave firmada por alguien cuya clave es válida para mi. Este sistema tal cual presenta serias deficiencias que vamos a tratar de solventar. Una clave se considera **válida** para un usuario si

- ha sido firmada por el propio usuario,
- ha sido firmada por varias entidades tales que todas juntas proporcionan suficiente nivel de confianza para el usuario

Anillos de confianza: confianza en usuarios

Cuatro niveles de confianza

unknown No se sabe nada sobre el dueño de la clave firmante. Suele ser el nivel por defecto asignado a una clave.

none Se sabe que el propietario firma otras claves de modo impropio.

marginal El propietario comprende las implicaciones de firmar una clave y valida las claves de forma correcta antes de firmarlas.

full El propietario comprende perfectamente las implicaciones de firmar una clave y su firma sobre una clave es tan buena como la nuestra.

Anillos de confianza: validez de la clave

Una clave se considera válida si

1. ha sido firmada
   - personalmente,
   - por un usuario con nivel de confianza pleno,
   - por tres usuarios con nivel de confianza marginal;
2. el camino que nos lleva desde nuestra clave hasta la clave firmada de cinco pasos o menos.

Los principales problemas vienen dados de la subjetividad implícita a la hora de conceder niveles de confianza, y a la gestión de los certificados caducados o revocados.

Estructuras jerárquicas

Hay una Autoridad de Certificación (CA) principal (raíz) que emite certificados a CAs subordinadas. Éstas a su vez certifican a usuarios y dispositivos en función del uso y validez de las claves.

- El certificado está firmado únicamente por el emisor.
- Suele incluir los datos necesarios para verificar la identidad del emisor a partir de la CA raíz.
- Su fortaleza depende de la política de la autoridad certificadora.
- Facilidad para buscar la información necesaria para verificar la información.

Autoridades de Certificación (CA) I

Objeto y Funciones

- Constituyen el elemento principal de la cadena de certificación.
- Técnicamente es un certificado auto-firmado.
- Si la CA se ve comprometida, todo el sistema queda invalidado.
- Sus funciones son
  - Actuar como una entidad de confianza.
  - Verificar la identidad de los solicitantes.
  - Emitir los certificados.

Autoridades de Certificación (CA) II

Públicas vs. Privadas

Privadas: Son creadas por una entidad concreta y delimitada para dar servicios de certificación dentro de esa misma entidad. No tienen validez fuera de dicha entidad. Hay libertad plena para diseñar la información que aparece en el certificado, así como para modificar y adaptar la política de certificación.

Públicas: Funcionan a un nivel mucho más amplio, por ejemplo en todo Internet. Suelen estar mantenidas y operadas por empresas que tienen este propósito como negocio principal, cobrando una tasa por ello, o por instituciones gubernamentales. El contenido del certificado, tipos de nombres y atributos, está mucho más limitado y debe adaptarse completamente a estándares definidos.

CA subordinadas

Objeto y funciones

- Mismas funciones que una CA:
  - Actuar como una entidad de confianza.
  - Verificar la identidad de los solicitantes.
  - Emitir los certificados.
- Técnicamente es un certificado firmado por una CA de nivel superior.
- Pueden crearse en base a criterios diversos:
  - Tipo de propietario: usuario, dispositivo, etc.
  - Uso del certificado: cifrado, firma, firma de CRL, firma de certificados, etc.
  - Geográficos: país, región, ciudad...

Autoridades de Registro (RA)

Objeto y funciones

- Dependen de una CA de nivel superior, ya sea raíz o subordinada.
- Sus funciones son:
  - Recibir las solicitudes de certificados.
  - Verificar la identidad del solicitante.
- No realizan tareas de firma digital.

Índice

Estructura

Un certificado X.509 consta de tres partes

`tbsCertificate` Este campo contiene los nombres del emisor y del propietario, la clave pública asociada al propietario, un periodo de validez e información adicional.

`signatureAlgorithm` Contiene el identificador del algoritmo criptográfico usado por la CA para firmar este certificado.

`signatureValue` Aquí encontramos la firma digital calculada sobre el campo `tbsCertificate` codificado en formato ASN.1 DER.

tbsCertificate

version Es un entero que admite los valores 0, 1 o 2 según la versión sea v1, v2 o v3.

serialNumber Entero positivo asignado por la CA al certificado. Único para cada certificado emitido por esa CA.

signature El identificador del algoritmo usado por la CA para firmar el certificado. Debe coincidir con el valor de `signatureAlgorithm`.

issuer Emisor identificado mediante un DN (nombre completo).

validity Intervalo de tiempo durante el cual la CA garantiza que mantendrá la información sobre el estado del certificado. Contiene dos fechas, `notBefore` y `notAfter`.

subject Propietario identificado mediante un DN.

subjectPublicKeyInfo Este campo contiene el valor de la clave pública y el identificador del algoritmo para el que esta clave se usa (RSA, DSA...)

issuerUniqueID No válido para v1. Contiene un identificador único del emisor. El objetivo es la posible reutilización del nombre del emisor. Su uso no se recomienda hoy en día.

ubjectUniqueID Idem para el propietario.

extensions Válidas para la v3.

signatureAlgorithm y signatureValue

El campo `signatureAlgorithm` contiene el identificador del algoritmo criptográfico usado por la CA para firmar este certificado. Aunque algunos algoritmos son considerados como aceptables dentro de los diferentes estándares, otros no considerados inicialmente pueden también ser utilizados.

El campo `signatureValue` contiene la firma digital calculada sobre `tbsCertificate` codificada según el estándar ASN.1 DER. El valor de la firma se codifica como una BIT STRING y se incluye en este campo.
Mediante la generación de esta firma, la CA certifica la validez de la información contenida en `tbsCertificate`. En particular, la CA certifica la correspondencia entre la clave pública y el propietario del certificado.

DN (Nombre completo)

El DN está formado por atributos. En principio cualquier tipo de atributo está permitido, pero las implementaciones deben reconocer al menos los siguientes:

- país,
- organización,
- unidad organizativa,
- clasificación del nombre completo,
- estado o provincia,
- nombre común,
- número de serie.

También deberían poder interpretar los siguientes:

- localidad,
- título,
- apellido,
- nombre,
- iniciales,
- pseudónimo,
- clasificador de la generación (“Jr.”, “3rd”, “IV”)

Extensiones I

Definidas para la v3, proporcionan métodos para asociar atributos adicionales del usuario o de la clave pública, así como gestionar relaciones entre CAs. Las extensiones deben clasificarse como

críticas el certificado debe ser rechazado si se encuentra una extensión crítica no reconocida o si su valor no puede ser procesado.

no críticas si no son reconocidas pueden ignorarse, pero si son reconocidas deben procesarse.

Aunque todas ellas son opcionales, en ocasiones la presencia o valores de alguna de ellas obliga o prohibe la presencia de otras. De igual forma, los certificados propiedad de las CAs deben contener algunas de las extensiones estándar.

Extensiones II

Extensiones estándar

- Identificador de la clave de la autoridad.
- Identificador de la clave del propietario.
- Uso de la clave.
- Políticas de los certificados.
- Asignaciones de la política.
- Nombre alternativo del propietario.
- Nombre alternativo del emisor.
- Atributos del directorio del propietario.
- Restricciones básicas.
- Restricciones del nombre.
- Política de restricciones.
- Uso de la clave extendido.
- Puntos de distribución de las CRL.
- CRL más actualizado.

Extensiones III

Extensiones privadas de Internet

- Información de acceso a la autoridad.
- Información de acceso al propietario.

Índice

Distribución y renovación

Distribución

Para solicitar un certificado a una CA suelen producirse los siguientes pasos.

1. Generación de pareja de claves pública/privada.
2. `Certificate Signing Request` conteniendo exclusivamente la clave pública.
3. Verificación de identidad.
4. Firma con la clave privada de la CA.
5. Devolución al propietario.

Renovación

Sigue los mismos pasos que en la distribución con la excepción de que la verificación de la identidad puede hacerse mediante un certificado existente del mismo propietario y próximo a caducar.

Verificación y revocación

Un certificado es correcto si:

- La firma es correcta.
- Nos encontramos dentro del periodo de validez del mismo.
- No ha sido revocado.

Cómo revocar un certificado

- `Certificate Revocation List`
- `Online Certificate Status Protocol`

Los motivos para revocar un certificado pueden ser variados:

- Descuido del propietario.
- Problema de seguridad en el equipo del propietario.
- Problema de seguridad en cualquier algoritmo de los usados en cifrado o firma.
- Problema de seguridad en algún equipo del emisor.

CRL

- Una CRL es una lista que contiene números de serie de certificados emitidos por una CA junto que la fecha a partir de la cual están revocados. Permite impedir el uso de un certificado antes de su fecha de caducidad.
- Puede almacenarse en el servidor de la CA que lo gestiona o en un servidor externo (recomendado).
- También es recomendable que el lugar al que acceden usuarios y dispositivos sea distinto que el empleado por la CA.
- Una CRL tiene también un periodo de validez. Una vez caducada, una nueva CRL debe ser generada y descargada por los distintos servidores.
- Las CRL no actúan en tiempo real. Las CRL son actualizadas y descargadas periódicamente.

Problemas de las CRL

- Cualquier operación entre la visión del certificado y la comprobación de la revocación puede provocar resultados inesperados.
- No se actualizan con suficiente rapidez.
- Caras de distribuir.
- Vulnerables a ataques DOS.
- Pueden invalidar una clave de forma retroactiva.
- ¿Cómo afecta a las CRL revocar certificados autofirmados?
- Al caducar una CRL se producen picos de carga en la red.
- Los usuarios deben elegir entre consumir ancho de banda y tener retrasos en el procesado, o perder algunas revocaciones. Depende del tiempo que tarden en actualizar las CRL.

OCSP

- Trata de evitar dos de los problemas de las `CRL`: el tamaño y el tiempo de vida.
- Se solicita al `OSCP Server` información concreta sobre la validez de un certificado.
- El `OSCP Server` comunica con la CA puntualmente o periódicamente. En este último caso los periodos son mucho más cortos que los empleados por las `CRL`
- Las respuestas pueden ser: *good*, *revoked*, *unknown*.
- El tipo *unknown* genera problemas semánticos.

¿Es necesaria la revocación?

Certificado de una CA comprometido: Revocación fácil de hallar. Se publicita y se informa de modo amplio.

Certificado de un servidor comprometido: Revocación difícil de hallar.

- Puede darse con cierta facilidad en servidores pobremente protegidos.
- El certificado se envía habitualmente en el *handshake*, por lo que no es necesaria la revocación. Basta con cambiar el certificado.

Certificado de usuario comprometido: El número de usuarios que pueden necesitar acceder al certificado de un usuario concreto es pequeño. Pueden enviarse avisos personalizados.

Alternativas

- Asociar certificados a tarjetas de crédito.
- Asociar certificados a cuentas en servidores.

Índice

Concepto

Un sello temporal confiable es un sello temporal emitido por una tercera entidad en la que se confía, que actúa como TSA (*Time Stamp Authority*). Se emplea para demostrar la existencia de ciertos datos antes de una fecha concreta. Al contrario que con las CAs, varias TSA son admisibles y deseables para unos datos concretos.

Podemos encontrarlos en los siguientes estándares:

- RFC 3161 . Sistema básico
- ANSI ASC X9.95, que añade integridad en los datos mediante un sello temporal de confianza.

TSA en RFC 3161

- Debe tener acceso a un reloj altamente confiable.
- Debe incluir un valor temporal confiable en cada sello temporal.
- Debe incluir un número de serie único en cada sello temporal.
- Debe producir un sello temporal al recibir una solicitud válida.
- Debe incluir en cada sello temporal un identificador que determine la política de seguridad empleada para producirlo.
- Debe producir el sello temporal sobre un hash de los datos, y nunca sobre los datos propios.
- Debe verificar que el la longitud del hash coincide con la descripción del hash empleado.
- No debe tratar de examinar los datos a los que se va a aplicar el sello temporal.
- No debe incluir en el sello temporal ninguna identificación del solicitante.
- Debe firmar cada sello temporal con una clave privada generada con éste único propósito, y debe tener esta propiedad de la clave indicada en el correspondiente certificado.
- Solo debe incluir en el sello temporal aquellas extensiones soportadas por la TSA.

Solicitud de un sello temporal

TimeStampReq

- version
- messageImprint
- reqPolicy
- nonce
- certReq
- extensions

messageImprint

- hashAlgorithm
- hashedMessage

Respuesta de un sello temporal I

TimeStampResp

- status PKIstatusInfo
- timeStampToken TimeStampToken

PKIstatusInfo

- status PKIStatus
- statusString PKIFreeText
- failInfo PKIFailureInfo

Respuesta de un sello temporal II

`PKIStatus`

- granted (0),
- grantedWithMods (1),
- rejection (2),
- waiting (3),
- revocationWarning (4),
- revocationNotification (5)

Respuesta de un sello temporal III

PKIFailureInfo

- badAlg (0),
- badRequest (2),
- badDataFormat (5),
- timeNotAvailable (14),
- unacceptedPolicy (15),
- unacceptedExtension (16),
- addInfoNotAvailable (17),
- systemFailure (25)

TimeStampToken

- contentType id-signedData
- content SignedData

Respuesta de un sello temporal IV

TSTInfo

- version INTEGER v1(1) ,
- policy TSAPolicyId,
- messageImprint
- serialNumber
- genTime GeneralizedTime,
- accuracy Accuracy OPTIONAL,
- ordering BOOLEAN DEFAULT FALSE,
- nonce INTEGER OPTIONAL,
- tsa [0] GeneralName OPTIONAL,
- extensions [1] IMPLICIT Extensions OPTIONAL

Solicitud, respuesta y verificación

Resumen de solicitud y respuesta.

- El solicitante envía un hash de los datos a sellar junto con información adicional necesaria para la creación del sello.
- La TSA concatena dicho hash con el sello temporal, vuelve a calcular el hash de dicha cadena, firma ese hash con su clave privada (cuya finalidad es exclusiva para firmar sellos temporales), y envía el valor de dicha firma junto con el sello temporal.
- El solicitante almacena conjuntamente sus datos, la firma y el sello.

Verificación

- Se calcula el hash de los datos. Este hash se concatena con el sello temporal.
- Se calcula el hash de la cadena anterior y se verifica mediante la clave pública del TSA si la firma es válida para dicha cadena.

Índice

Data hiding I

Data (o *information*) *hiding* es un término general que engloba a un amplio espectro de problemas que comprenden técnicas para incrustar mensajes en otro contenido.

Marcas de agua Es la práctica que consiste en alterar un trabajo de manera imperceptible para incrustar un mensaje sobre dicho trabajo.

Esteganografía Es la práctica que consiste en alterar un trabajo de manera indetectable para incrustar un mensaje secreto.

Data hiding II

Cuadro: Tipos de Data Hiding

	Mensaje dependiente	Mensaje independiente
Existencia oculta	Marcas ocultas	Esteganografía
Existencia conocida	Marcas de agua	Otros tipos

- Detección de filtraciones.
- Espíonaje.
- Contenido de museos / agencias.
- Señal horaria en señales de radio.

Data hiding III

Un esquema de ocultación de información consta de cinco elementos:

- Un conjunto $\mathcal{C}$ de posibles trabajos,
- un conjunto $\mathcal{M}$ de posibles mensajes,
- un conjunto $\mathcal{K}$ de posibles claves,
- una función $\text{inc} : \mathcal{C} \times \mathcal{M} \times \mathcal{K} \to \mathcal{C}$,
- una función $\text{rec} : \mathcal{C} \times \mathcal{C} \times \mathcal{K} \to \mathcal{M}$

satisfaciendo para cada $c \in \mathcal{C}$, $m \in \mathcal{M}$ y $k \in \mathcal{K}$,

$$\text{rec}(\text{inc}(c, m, k), c, k) = m.$$

Data hiding IV

La función de recuperación puede ser independiente del trabajo original, es decir,

$$\mathrm{rec}(d, c_1, k) = \mathrm{rec}(d, c_2, k)$$

para cualesquiera $c_1, c_2, d \in \mathcal{C}$ y $k \in \mathcal{K}$. En este caso podemos ver $\mathrm{rec} : \mathcal{C} \times \mathcal{K} \to \mathcal{M}$, satisfaciendo para cada $c \in \mathcal{C}$, $m \in \mathcal{M}$ y $k \in \mathcal{K}$,

$$\mathrm{rec}(\mathrm{inc}(c, m, k), k) = m.$$

Data hiding V

- Normalmente el uso de la clave se deja a la incorporación de técnicas criptográficas, por lo que prescindiremos habitualmente de ellas.
- En el mundo digital $\mathcal{C} = \mathcal{A}^n$ y $\mathcal{M} = \mathcal{A}^k$ para cierto alfabeto finito $\mathcal{A}$, que normalmente es $\mathcal{A} = \mathbb{Z}_q$ o $\mathcal{A} = \mathbb{F}_{2^t}$. De esta forma

$$\text{inc} : \mathcal{A}^n \times \mathcal{A}^k \to \mathcal{A}^n$$

y

$$\text{rec} : \mathcal{A}^n \times \mathcal{A}^n \to \mathcal{A}^k \quad \text{o} \quad \text{rec} : \mathcal{A}^n \to \mathcal{A}^k$$

Data hiding VI

En $\mathcal{A}^n$ consideraremos la distancia de Hamming: para cualesquiera $x = (x_0, \ldots, x_{n-1}), y = (y_0, \ldots, y_{n-1}) \in \mathcal{A}^n$,

$$\mathrm{d_H}(x, y) = \#\{0 \leq i \leq n-1 \mid x_i \neq y_i\},$$

es decir, el número de posiciones en que x e y difieren.

Si $\mathcal{A} = \mathbb{F}_q$, se define el peso de Hamming de $x = (x_0, \ldots, x_{n-1}) \in \mathbb{F}_q^n$ como

$$\mathrm{w_H}(x) = \mathrm{d_H}(x, 0) = \#\{0 \leq i \leq n-1 \mid x_i \neq 0\}.$$

Observemos que

$$\mathrm{d_H}(x, y) = \mathrm{w_H}(x - y).$$

Watermarks: Antecedentes I

Marcas de agua

- Aclarado de papel: Fraude en emisión de monedas
- Información destacando letras de un libro.
- Código de identificación en grabaciones musicales insertando muescas a muy baja frecuencia (1 kHz).
- Marcas de agua digitales: sistemas de protección anticopia.

Watermarks: Antecedentes II

Esteganografía

- Tatuajes en la cabeza.
- Tablas de madera y cera.
- Accesorios de indumentaria (sobre todo mujeres).
- Tamaño de las letras.
- Acrósticos (esteganografía lingüística).
- Cambios en la tipografía (Francis Bacon).
- Esteganografía digital.

Watermarks: Características I

Las marcas de agua buscan asociar información sobre un trabajo al mismo. ¿Por qué no usar otras técnicas más sencillas para el mismo fin?

Porque...

- son imperceptibles,
- son inseparables,
- sufren el mismo tipo de transformaciones que el trabajo en el que están incrustadas.

Watermarks: Aplicaciones I

Seguimiento de emisiones multimedia: Identificar cuándo y dónde se han emitido los trabajos reconociéndolos por las marcas de agua incrustadas en ellos. Otras soluciones, como bases de datos de contenidos o analizadores semánticos son tremendamente costosos e ineficientes.

Identificación del propietario: Incrustación de la identidad del poseedor de los derechos del trabajo como una marca de agua. Los métodos legalmente establecidos son fácilmente eliminables voluntaria o involuntariamente.

Prueba de propiedad: Uso de marcas de agua para proporcionar evidencias en caso de disputas sobre la propiedad. Complementa a la aplicación anterior. Otros sistemas, normalmente basados en la creación de repositorios centralizados, son caros tanto para el proveedor como para el usuario.

Seguimiento de las transacciones: Uso de las marcas de agua para identificar personas que obtienen el contenido legalmente pero lo distribuyen ilegalmente. Normalmente se logra insertando marcas de agua con un número de serie propio para cada copia.

Watermarks: Aplicaciones II

Autenticación del contenido: Incrustar una firma digital o un MAC para más tarde comprobar que el contenido no ha sido alterado. Sin la incrustación pueden ser fácilmente borradas. Esto incluye, por ejemplo, vídeos de vigilancia. La aplicación de filtros a contenido multimedia puede ser más fácilmente detectable si existen marcas de agua.

Control de copia: La criptografía es la mejor herramienta para proteger el acceso a contenido registrado. Sin embargo, una vez descifrado es fácilmente duplicable. Uso de marcas de agua puede indicar al equipo grabador qué contenido no debe ser grabado.

Control de dispositivos: Uso de marcas de agua para que los dispositivos reacciones ante el contenido mostrado.

Mejoras heredadas: Uso de marcas de agua para mejorar la funcionalidad de sistemas existentes.

Watermarks: Propiedades I

La idoneidad de un sistema de marcas de agua para una aplicación concreta puede ser juzgada en términos de las siguientes propiedades.

Incrustación efectiva: Probabilidad de una incrustación satisfactoria dentro de un trabajo aleatoriamente seleccionado. Lo deseable es el 100 %, aunque el coste de garantizarla puede ser incompatible con otras propiedades. Puede ser calculada de forma analítica o mediante una aproximación estadística.

Fidelidad: La calidad en la percepción en el contenido con la marca de agua incrustada. Puede depender de la posible degradación de la señal desde que se emite hasta que llega al receptor.

Datos cargados: Cantidad de información que una marca de agua puede incluir. Se mide en bits, y depende del uso. Por ejemplo, el control de copia no requiere más de 8 bits cada 10 segundos de música o 5 minutos de vídeo. El seguimiento de emisiones requiere 24 bits en el primer segundo.

Watermarks: Propiedades II

Detección ciega o informada: Si el recuperador puede detectar una marca de agua en un trabajo sin tener información adicional o si necesita alguna información relacionada con la versión original del trabajo. La aplicación puede ser crucial a la hora de establecer el tipo de marca de agua.

Tasa de falsos positivos: Frecuencia con la debemos esperar una recuperación falsa de una marca de agua en un contenido que carece de ella. Como variable aleatoria podemos considerar tanto los trabajos como los mensajes. El diseño con mayor o menor tasa depende, como siempre, de la aplicación.

Robustez: La habilidad de la marca de agua para sobrevivir al procesado normal del contenido. Hay ocasiones en que lo interesante es la ausencia de esta propiedad.

Seguridad: Habilidad de la marca de agua para resistir ataques hostiles. Pueden ser de tres tipos: Borrados, incrustación (activos) o detección (pasivo) no autorizados.

Modificación y múltiples marcas de agua: Posibilidad de cambiar la marca de agua incrustada o incluir varias marcas.

Coste: Coste computacional del incrustador y del recuperador.

Watermarks: Observaciones I

- Las propiedades requeridas de una marca de agua dependen de la aplicación.
- Benchmarking es un medio razonable para comparar sistemas de marcas de agua. Sin embargo, ninguna referencia concreta es probable que sea relevante para todas las aplicaciones.
- Los sistemas de marcas de agua deben probarse con conjuntos grandes de datos.
- Si un sistema de marca de agua se mejora de manera que tenga un mejor rendimiento en una propiedad, a menudo produce mejoras de rendimiento en otras propiedades.

Esteganografía: Características I

- La esteganografía es una herramienta para la privacidad.
- La esteganografía se considera rota si simplemente se detecta la mera existencia del mensaje.
- La indetectabilidad suele garantizarse con mensajes suficientemente cortos.
- Los sistemas esteganográficos deben comprobarse mediante ataques ciegos y dirigidos, realizados sobre conjuntos de datos grandes y diversos.
- Algunas aplicaciones de la esteganografía pueden obtenerse con otras tecnologías. La principal ventaja radica en que la propia existencia de la comunicación se mantiene en secreto.

Esteganografía: Aplicaciones I

- Espionaje.
- Comunicaciones encubiertas entre disidentes.
- Comunicaciones encubiertas entre criminales.

Esteganografía: Propiedades I

Las propiedades comentadas sobre las marcas de agua tienen la siguiente interpretación para sistemas esteganográficos.

Incrustación efectiva: No es relevante.

Fidelidad: No necesariamente relevante.

Capacidad esteganográfica: Es el equivalente a los datos cargados. Cantidad de bits que pueden ser ocultados en un trabajo de tal manera que la probabilidad de detección por parte de un adversario sea despreciable.

Capacidad de incrustación: Máximo número de bits que pueden ser ocultados en un trabajo dado.

Eficiencia de la incrustación: Número de bits del mensaje incrustados por unidad de distorsión.

Recuperación ciega o informada: Depende de si el receptor dispone o no de una copia original del trabajo.

Estegoanálisis del sistema: Se refiere a ataques que se basan en debilidades de la implementación en lugar de en la naturaleza de la incrustación.

Estegoanálisis ciego: Métodos de detección independientes del sistema esteganográfico usado.

Esteganografía: Propiedades II

Estegoanálisis dirigido: Métodos de ataque diseñados contra sistemas esteganográficos concretos.

Indetectabilidad estadística: Probabilidad de detectar un trabajo incrustado basándose en las distribuciones de probabilidad de trabajos con y sin incrustación.

Tasa de falsa alarma: Probabilidad de que un algoritmo de análisis detecte la presencia de de un mensaje incrustado cuando éste no está.

Robustez: Normalmente este paso se obvia en la esteganografía, pues en la actualidad los sistemas digitales carecen de degradación.

Seguridad: Se suele evaluar en términos de ataques pasivos.

Modificación y múltiples mensajes: No es aplicable.

Coste: No es relevante.

Estegosistemas: Reglas de selección

Normalmente no se emplea todo el mensaje/trabajo para realizar la incrustación:

- Para marcas de agua la incrustación debe realizarse en aquellas zonas que minimicen la percepción de las mismas y a la vez garanticen la utilidad de las marcas.
- En la esteganografía, la incrustación debe realizarse en aquellos lugares en los que sea más difícil de detectar.

El caso más intuitivo es la incrustación de datos en imágenes. Pensemos en una imagen presentada como una colección de píxeles, cada uno indicando un tono de gris mediante D bits, es decir, cada píxel es un número en el rango $\{0, \ldots, 2^D - 1\}$. Para ocultar datos en la imagen debemos decidir cuántos bits vamos a cambiar en cada píxel. Dicha elección nos conducirá a decidir qué píxeles podemos cambiar y cuáles no.

El mensaje o trabajo recibe el nombre en inglés de *cover work* o *cover object*. La parte seleccionada para incrustar los datos recibe el nombre de *cover sequence* o *cover vector*. Las reglas para elegirlos reciben el nombre de *selection rules*.

Estegosistemas: Parámetros I

- Longitud del trabajo n.
- Capacidad de incrustación k.
- Radio de incrustación

$$\rho = \max\left\{ \mathrm{d_H}(c, \mathrm{inc}(c, m)) \mid c \in \mathcal{A}^n, m \in \mathcal{A}^k \right\}.$$

- Número medio de cambios en cada incrustación

$$R_a = \frac{1}{q^{kn}} \sum_{\substack{c \in \mathcal{A}^n \\ m \in \mathcal{A}^k}} \mathrm{d_H}(c, \mathrm{inc}(c, m))$$

- Carga relativa $\alpha = \frac{k}{n}$, y carga binaria relativa (o tasa de incrustación) $E = \frac{k}{n} \log_2(q)$.
- Tasa de cambio (o distorsión media) $c = \frac{R_a}{n}$.
- Eficiencia de incrustación $e = \frac{k}{R_a}$ y eficiencia de incrustación mínima $\underline{e} = \frac{k}{\rho}$.

LSB modulation

La entrada es una imagen dada como una matriz de vectores en $\mathbb{F}_2^D$, es decir, una matriz de cadenas de D bits.

La selección se realiza en dos fases:

- Seleccionamos qué píxeles vamos a emplear, lo que nos da una lista $s = (s_1, \ldots, s_n)$. Cada $s_i = (b_{i,0}, \ldots, b_{i,D-1}) \in \mathbb{F}_2^D$ equivale al número escrito en binario $s_0 = \sum_{j=0}^{D-1} b_{i,j} 2^j$.
- La *cover sequence* se obtiene concatenando los t bits menos significativos, es decir,

$$b_{1,0} \ldots b_{1,t-1} b_{2,0} \ldots b_{2,t-1} \ldots b_{n,0} \ldots b_{n,t-1}$$

Imágenes JPEG

JPEG

- La imagen se divide en bloque de 8×8 píxeles.
- Se aplica la Transformada de Coseno Discreta (DCT) a dichos bloques, lo que proporciona el espectro de frecuencia espacial.
- Los datos resultantes se comprimen con el conocido como algoritmo sin pérdida de Huffmann.

Data hiding en JPEG

- Se seleccionan algunos de los coeficientes DCT.
- El cover vector consiste en la sucesión de los bits menos significativos (LSB) de los coeficientes anteriores.

Índice

Códigos Correctores de Errores I

ECC

- Fijamos como alfabeto un cuerpo finito $\mathbb{F}_q$.
- Un $(n, M)_q$-código es un subconjunto $\mathcal{C} \subseteq \mathbb{F}_q^n$ de cardinal M. Decimos que n es la longitud del código.
- La distancia mínima de $\mathcal{C}$ se define como

$$\mathrm{d}(\mathcal{C}) = \mathrm{mín}\{\mathrm{d_H}(x, y) \mid x, y \in \mathcal{C}, x \neq y\}$$

- Un algoritmo de decodificación es una aplicación $\mathrm{dec} : \mathbb{F}_q^n \to \mathcal{C}$ tal que para cualquier $x \in \mathbb{F}_q^n$, $\mathrm{d_H}(x, \mathcal{C}) = \mathrm{d_H}(x, \mathrm{dec}(x))$.

Códigos Correctores de Errores II

Capacidad de corrección

- Si transmitimos $c \in \mathcal{C}$ a traves de un canal con ruido, recibiremos $c + e$ con e cierto error. Si $2\,\mathrm{w_H}(e) < \mathrm{d}(\mathcal{C})$, entonces $\mathrm{dec}(c + e) = c$.
- $t = \left\lfloor \frac{\mathrm{d}(\mathcal{C})-1}{2} \right\rfloor$ es la capacidad de corrección del código $\mathcal{C}$.
- $B(c, t) = \{x \in \mathbb{F}_q^n \mid \mathrm{d_H}(x, c) \leq t\}$. Si $c \in \mathcal{C}$ y $x \in B(c, t)$, entonces $\mathrm{dec}(x) = c$. El cardinal de estas "bolas" es fácil de calcular: $\#B(c, t) = \sum_{i=0}^{t} \binom{n}{i}(q-1)^i$.
- La cota de Hamming establece

$$M \sum_{i=0}^{t} \binom{n}{i} (q-1)^i \leq q^n.$$

- $\mathcal{C}$ se dice perfecto si se alcanza la cota de Hamming. En este caso todo elemento de $\mathbb{F}_q^n$ se decodifica a un elemento de $\mathcal{C}$.

Enlazando ECC y estegosistemas I

- Sea $\mathcal{S} = (\text{inc}, \text{rec})$ un sistema estaganográfico donde $\text{inc} : \mathbb{F}_q^n \times \mathbb{F}_q^k \to \mathbb{F}_q^n$ y $\text{rec} : \mathbb{F}_q^n \to \mathbb{F}_q^k$.
- Para cada $m \in \mathbb{F}_q^k$, sea el código $\mathcal{C}_m = \{x \in \mathbb{F}_q^n \mid \text{rec}(x) = m\}$.
- La familia $\{\mathcal{C}_m \mid m \in \mathbb{F}_q^k\}$ es una partición de $\mathbb{F}_q^n$.
- Para cada $m \in \mathbb{F}_q^k$, la aplicación $\text{dec}_m : \mathbb{F}_q^n \to \mathcal{C}_m$ definida por $\text{dec}_m(x) = \text{inc}(x, m)$ es un algoritmo de decodificación.

- Sea $\{\mathcal{C}_m \mid m \in \mathbb{F}_q^k\}$ una familia de códigos que constituyen una partición de $\mathbb{F}_q^n$.
- Para cada $m \in \mathbb{F}_q^k$ denotamos $\text{dec}_m : \mathbb{F}_q^n \to \mathcal{C}_m$ al correspondiente algoritmo de decodificación.
- Sea $\text{inc} : \mathbb{F}_q^n \times \mathbb{F}_q^k \to \mathbb{F}_q^n$ la aplicación definida por $\text{inc}(x, m) = \text{dec}_m(x)$.
- Sea $\text{rec} : \mathbb{F}_q^n \to \mathbb{F}_q^k$ la aplicación definida por $\text{rec}(x) = m$ si $x \in \mathcal{C}_m$.
- La pareja $\mathcal{S} = (\text{inc}, \text{rec})$ es un estegosistema.

Enlazando ECC y estegosistemas II

En el diseño de métodos de ocultación de información debemos centrar los esfuerzos en la función de recuperación, ya que es la que determina el sistema completo.

Códigos lineales y afines I

Definiciones

- Un $[n, k]_q$–código lineal es un subespacio vectorial $\mathcal{C} \leq \mathbb{F}_q^n$ de dimensión k.
- Un $[n, k]_q$–código afín es un subespacio afín de $\mathbb{F}_q^n$ de dimensión k, es decir, un subconjunto de la forma $z + \mathcal{C}$ donde $\mathcal{C}$ es un $[n, k]_q$–código lineal.

Lema

Para cualesquiera $x, y, z \in \mathbb{F}_q^n$, $\mathrm{d_H}(x, y) = \mathrm{d_H}(x + z, y + z)$.

Proposición

Sea $\mathrm{dec} : \mathbb{F}_q^n \to \mathcal{C}$ *un algoritmo de decodificación para un código lineal* $\mathcal{C}$. *La aplicación* $x \mapsto x + \mathrm{dec}(x - z)$ *es un algoritmo de decodificación para el código afín* $z + \mathcal{C}$.

Códigos lineales y afines II

Sea $\mathcal{C}$ un $[n.k]_q$–código lineal.

- Una matrix $G \in \mathcal{M}_{k\times n}(\mathbb{F}_q)$ tal que

$$\mathcal{C} = \{mG \mid m \in \mathbb{F}_q^k\}$$

  recibe el nombre de matrix generadora o *encoder*.

- Una matrix $H \in \mathcal{M}_{(n-k)\times n}(\mathbb{F}_q)$ tal que

$$\mathcal{C} = \{x \in \mathbb{F}_q^n \mid xH^{\mathrm{T}} = 0\}$$

  recibe el nombre de matrix (verificadora) de paridad o *parity check matrix*.

Convertir matrices generadoras en matrices de paridad es el mismo proceso que convertir sistemas de generadores (bases) en ecuaciones cartesianas y viceversa.

Códigos lineales y afines III

Códigos de Hamming

Una familia de códigos lineales adecuada para esta tarea son los códigos de Hamming. Los códigos de Hamming binarios son $[2^r-1, 2^r-1-r]_2$–códigos lineales cuya matriz de paridad tiene por columnas los números $1, 2, \ldots, 2^r-1$ escritos en binario. Por ejemplo, el $[7,4]$–código de Hamming tiene por matriz de paridad

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Estos códigos tienen distancia 3 y son muy sencillos de decodificar.

Códigos lineales y afines IV

Síndrome

Sea H una matrix de paridad de un $[n,k]_q$–código lineal $\mathcal{C}$. Se define el síndrome como la aplicación

$$\begin{aligned} \operatorname{syn} : \mathbb{F}_q^n &\to \mathbb{F}_q^{n-k} \\ x &\mapsto \operatorname{syn}(x) = xH^{\mathrm{T}} \end{aligned}$$

Observemos que si transmitimos una palabra $c \in \mathcal{C}$ y recibimos $y = c + e \in \mathbb{F}_q^n$, tenemos que

$$\operatorname{syn}(y) = (c+e)H^{\mathrm{T}} = cH^{\mathrm{T}} + eH^{\mathrm{T}} = \operatorname{syn}(e).$$

Códigos lineales y afines V

Decodificación por síndrome

La decodificación por síndrome consiste en generar una lista

Cuadro: Síndromes

Síndrome	Error
s_1	e_1
s_2	e_2
$\vdots$	$\vdots$

donde la columna error contiene de el conjunto $\{x \in \mathbb{F}_q^n \mid \text{syn}(x) = s_i\}$ un elemento de peso mínimo, y para cada palabra recibida $y \in \mathbb{F}_q^n$

$$\text{dec}(y) = e_i \text{ si } \text{syn}(y) = s_i.$$

Índice

Índice

Seguridad física

La seguridad física es el proceso de proteger el contenido de una instalación mediante un elemento físico que pueda resistir la penetración por parte de intrusos.

Consta de cuatro mecanismos:

- **Disuasión:** Creación de una atmósfera destinada a asustar a posibles intrusos.
- **Prevención:** Incluye cortafuegos, zonas desmilitarizadas, llaves, tarjas de acceso, identificación biométrica, etc.
- **Detección:** Emisión de señales que indiquen la presencia de un intruso. Pueden ser en tiempo real o mediante un análisis posterior.
- **Respuesta:** En dos sentidos, detener la intrusión y evitar intrusiones futuras.

Seguridad teórica

También conocida como seguridad por oscuridad, se basa en la creencia de que un objeto es seguro en tanto en cuanto nadie conoce su existencia más allá de un núcleo central y reducido de entidades. La seguridad se asienta en la presunción de que sólo aquellas entidades implicadas, que son confiables, pueden usar el sistema asegurado, y nadie más necesita conocerlo. Se basa por tanto en la confianza en las entidades.

Aunque es un estado de seguridad falaz y que ya ha demostrado su ineficacia, sigue siendo empleado en muchos desarrolladores de software y agencias gubernamentales.

Redes de ordenadores

Una red de ordenadores es un conjunto de ordenadores conectados de forma distribuida. Esta conexión puede ser

- fuerte: los ordenadores comparten una gran cantidad de recursos de un ordenador central,
- débil: sólo comparten unos pocos recursos necesarios para que la red funcione.

La seguridad de una red se basa en crear un entorno en el que una red funcione de manera segura, incluyendo todos sus recursos como son el almacenamiento y tránsito de datos y sus usuarios.

Seguridad de ordenadores y de la información

Seguridad de ordenadores

Creación de un entorno encaminado a garantizar el uso seguro de ordenadores por parte de usuarios. Incluye aspectos éticos y el desarrollo de protocolos, entre los que se incluyen los protocolos criptográficos.

Seguridad de la información

Es un nivel superior a los anteriores. Se trata de crear el entorno en el que la información y los datos permanezcan seguros. Involucra varias disciplinas, incluyendo informática, empresarial, legislativa, etc.

Redes seguras

Un recurso es seguro si está protegido contra accesos no autorizados tanto internos como externos. Los recursos son objetos tangibles o intangibles, es decir, recursos hardware o datos almacenados y en tránsito.

Hardware

Proteger recursos hardware incluye:

- Los interfaces de comunicación como teclados, ratone, pantallas táctiles, lápices, etc.
- Cortafuegos, concentradores, conmutadores, entutadores, pasarelas, etc. vulnerables a hackers.
- Canales de comunicación para prevenir espionaje e interceptación de comunicaciones.

Software

Incluye la protección de firmware, sistemas operativos, protocolos a nivel de servidor, navegadores, aplicaciones, y los datos que generan y gestionan.

Índice

Formas de protección

Los primeros elementos para prevenir accesos no autorizados son establecer medidas para controlar dicho acceso y determinar la autenticidad de las entidades.

Otras características de seguridad como la confidencialidad, integridad, secreto y no repudio son también necesarias en la seguridad en redes. Sin embargo su garantía se deja a las técnicas estudiadas anteriormente, como el uso de criptosistemas, funciones hash, protocolos de firma digital y esteganografía.

Control de accesos I

Hardware

- Terminal de acceso. Hoy en día incluyen identificación, derechos de acceso, puntos de control, y comunicación con anfitriones. Pueden incluir verificación por huellas dactilares y sensores anti-rotura en tiempo real, y funcionar en línea o en solitario.
- Monitorización visual. Incluye el uso de imágenes y audio en tiempo real, así como tecnologías de localización tipo GPS.
- Tarjetas de identificación. Usando tecnologías diversas son un instrumento físico útil para gestionar los accesos.
- Identificación biométrica.
- Vídeo-vigilancia. El análisis de imágenes de vídeo en circuito cerrado permite hoy en día una respuesta casi en tiempo real.

Control de accesos II

Software

En los *Puntos de Acceso* las actividades personales son monitorizadas por una aplicación que almacena aquellos eventos relacionados con el acceso.
En el modo remoto se emplean técnicas de reconexión automática o presentación de informes regularmente en terminales conectados de varias maneras.

Autenticación I

Cosas que uno sabe

Suelen basarse en el uso de nombres de usuario acompañados de contraseñas o frases contraseña. Debemos tener presente la premisa "fácil de recordar es fácil de atacar". Por otra parte, una contraseña difícil de recordar será probablemente apuntada, por lo que pasa a ser algo que uso sabe a algo que uno tiene.

Cosas que uno tiene

- Tarjetas de identificación. Suelen emplearse coordinadas con otros medios, pues quien consigue la tarjeta puede suplantar la identidad del propietario.
- Localización física. Los medios más empleados son las direcciones hardware de los dispositivos de red o las direcciones software. Sólo autentifican el lugar de conexión.

Autenticación II

Técnicas biométricas

Retina, huellas dactilares, etc. Comparan la imagen obtenida mediante un escáner o similar con una base de datos almacenada en el sistema.

Índice

Conexiones por cable I

Hilo de cobre y par trenzado

Los cables de cobre han sido los tradicionalmente más empleados para las conexiones por su baja resistencia a las corrientes eléctricas. Son muy sensibles a las interferencias electromagnéticas, por lo que deben ir aislados.
El par trenzado consiste en un par de hilos de cobre aislados que se unen girando cada uno sobre el otro repetidamente. El giro los hace más resistente a interferencias electromagnéticas externas y evitan que ellos mismos las generen. Suele unirse más de un par trenzado de hilos en un mismo cable, como en los estándares `RJ-11` y `RJ-45`.

Cable coaxial

Consiste en un núcleo central, habitualmente un hilo macizo de cobre, rodeado de una capa de aislamiento, a su vez rodeada de una malla externa conductora. Todo el conjunto está protegido por una capa externa aislante. Los hay de varios tamaños, y suelen transportar señales de alta frecuencia a través de largas distancias.

Conexiones por cable II

Fibra óptica

Es un tubo de cristal y plástico que conduce señales ópticas. Permite transmitir con un gran ancho de banda, y es mínimamente sensible a interferencias electromagnéticas. Por otra parte su sincronización es mucho más compleja que en los cables de cobre. La luz circula generada por diodos de emisión de luz (LED) o diodos de inyección de láser (ILD).

Conexiones inalámbricas I

Alcance

- Redes restringidas próximas. Incluyen las LAN que abarcan tanto conexiones inalámbricas como conexiones cableadas.
- Redes intermedias/extendidas. Conectan dos LAN previamente funcionales mediante una conexión inalámbrica. Suelen emplearse para conectar, por ejemplo, dos edificios cercanos.
- Redes móviles. Suelen conectar una unidad móvil con otra fija mediante tecnologías inalámbricas como satélites o telefonía móvil.

Conexiones inalámbricas II

Tipos

- Infrarrojos. Se envían pulsos de luz infrarroja. Funcionan en tanto en cuanto no haya objetos que bloqueen el paso de la luz. Sirven para distancias cortas, del orden de unos 30 metros máximo. Soporta anchos de banda de hasta 10 Mbps.
- Radio de alta frecuencia. Permite distancias más largas que los infrarrojos. Puede sortear obstáculos debido al carácter de onda de las emisiones de radio.
- Microondas. Son emisiones de mayor frecuencia que las anteriores. En este caso la señal se enfoca directamente a un receptor, por lo que es necesario alinear perfectamente emisor y receptor. Pueden ser de corta o larga distancia (satélites).
- Láser. Su uso, fuera de la fibra óptica, permite alcanzar largas distancias siempre que no haya obstáculos intermedios. Como en el caso de las microondas se suelen emplear repetidores.

Índice

Modelo TCP/IP

Modelo de comunicación basado en 5 capas:

Capa	Unidad de reparto	Protocolos
Aplicación	Mensaje	FTP, NSP, SMTP, SNMP, HTTP, telnet, NFS, DNS, DHCP, BOOTP, etc.
Transporte	Segmento	TCP, UDP
Red	Datagrama	IP, ICMP, IGMP
Enlace de datos	Marco	
Física	Flujo de bits	Controladores de tarjetas de red

Más simple que el modelo OSI y más usado.

TCP/IP: Aplicación

Proporciona una interfaz de usuario con recursos ricos en funciones de aplicación. Cada protocolo de aplicación añade una cabecera específica al mensaje siguiendo el esquema

Cabecera del protocolo de aplicación	Mensaje

TCP/IP: Transporte I

Su principal objetivo es el transporte de los mensajes generados en la capa de aplicación entre el anfitrión y el servidor.

TCP

Es un servicio orientado a la conexión. Garantiza el reparto de los paquetes generados en la capa de aplicación a su destino mediante dos mecanismos:

- Control de congestión.
- Control de flujo.

Dirección de la fuente	Dirección del destino
Número de secuencia	Número de contestación
Información de control adicional	
Datos	

TCP/IP: Transporte II

UDP

Proporciona un servicio de conexión austero sin contestación ni garantías de reparto. Es más eficiente, se emplea en emisiones de video y música.

Dirección de la fuente	Dirección del destino
Información de control adicional	UDP Checksums
Datos	

TCP/IP: Red

Esta capa mueve paquetes, que ahora se llaman datagramas, de un enrutador a otro a través de un camino entre el anfitrión fuente y el anfitrión destino. Entre los protocolos que comprende están IP (Internet Protocol), ICMP (Internet Control Message Protocol) e IGMP (Internet Group Management Protocol).

IP

Este protocolo emplea la información aparecida en la cabecera de la capa de transporte, incluyendo los números de puerto y las direcciones IP de fuente y destino, para mover los datagramas entre enrutadores a lo largo de la red. La mejores rutas se obtienen mediante algoritmos de enrutado. La estructura de un datagrama IP es la siguiente:

Información de control adicional	Número de puerto de la fuente	Número de puerto del destino	Datos

Las direcciones IP pueden ser de 32 bits, IPv4, o de 64 bits, IPv6.

Consecuencias

En el momento del desarrollo inicial de los protocolos de comunicaciones, la seguridad no era una necesidad. Se obvió la implementación de medidas que garantizaran autenticidad, integridad, confidencialidad, etc.

Vista una red como un gran sistema con elementos distribuidos, debemos de asegurar dos aspectos:

- Nodos. Se logra mediante métodos de control de accesos, cortafuegos, etc.
- Tráfico. Mecanismos criptográficos.

Estos últimos son los que vamos a describir a continuación.

IPsec

Detallada en [KS05]. Se basa en dos protocolos

- **AH** Authentication Header Protocol proporciona autenticidad e integridad.
- **ESP** Encapsulated System Payload Protocol proporciona confidencialidad.

Se implementan en la capa de red.

- Como parte integral de IPv6.
- Como una interfaz entre la capa IP y la capa de red.
- Como un motor criptográfico separado.

Asociaciones de seguridad (SA)

Es un canal seguro unidireccional que proporciona autenticidad (AH) o confidencialidad (ESP). Si fuesen necesarias ambas características, se obtendría concatenando dos SA en el siguiente orden:

$$M \rightarrow \mathrm{SA}_{\mathtt{ESP}} \rightarrow \mathrm{SA}_{\mathtt{AU}}$$

Cada SA se identifica mediante una tripleta: dirección IP del destino, índice de parámetros de seguridad y protocolo seguro empleado (ESP o AH).

Una SA puede actuar de dos modos

Túnel: El datagrama es encapsulado en un nuevo datagrama con una nueva cabecera.

Transporte: Añade datos adicionales a la cabecera previa del datagrama existente.

AH I

El protocolo AH proporciona autenticidad de los datagramas IP. Incluye la siguiente cabecera que se coloca inmediatamente después de la cabecera IP

Next Header	Payload Length	Reserved
Security Parameter Index (SPI)		
Sequence Number Field		
Authentication Data		

Se emplea usualmente en modo transporte.

Next Header

8 bits, indica el tipo de la próxima carga siguiendo al AH.

AH II

Payload Length

8 bits, indica la longitud del AH en palabras de 32 bits.

SPI

32 bits, identifica de manera única la SA.

Sequence Number Field

Indica la posición del datagrama dentro del flujo de paquetes enviados a través de la SA.

AH III

Authentication Data

Contiene el ICV (Integrity Check Value) usado por el receptor para verificar la autenticidad del paquete. Es un MAC generado de una de las siguientes formas:

- Tipo `CBC-MAC` basado en criptosistemas simétricos.
- Tipo `HMAC`.
- Firma digital.

Se calcula sobre las partes inmutables del paquete.
Evitan ataques de repetición mediante el Sequence Number Field. Una vez enviados los posibles 2^{32} paquetes, la SA se cierra y se inicia una nueva SA.

ESP I

Garantiza confidencialidad. Puede funcionar en modo túnel o transporte. Actúa sobre un paquete del tipo

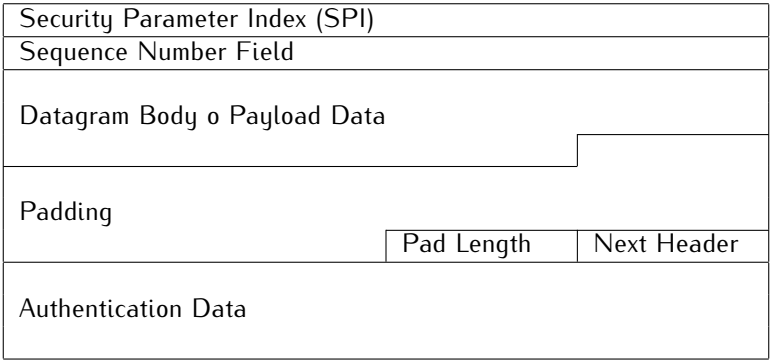
IP Header	Datagram Payload

y lo transforma en

Transporte	IP Header	ESP Header	Datagram Payload	ESP Footer
Túnel	New IP Header	ESP Header	IP Header	Datagram Payload	ESP Footer

ESP II

Paquete ESP



ESP III

SPI

32 bits, identifica de manera única la SA.

Sequence Number Field

32 bits, indica la posición del datagrama dentro del flujo de paquetes enviados a través de la SA. Empieza en 1, y cuando se alcanza 2^{32} se cierra la SA y se inicia otra nueva.

Datagram Body

Datos contenidos en el paquete original.

Padding

Necesario para ajustar el tamaño de los datos cifrados a un múltiplo de palabras de 32 bits. Nunca será mayor de 255 bytes.

ESP IV

Pad Length

Número de bytes en el campo Padding.

Next Header

8 bits, indica el tipo de datos en el campo Datagram Body.

Authentication Data

Contiene el MAC (o ICV) de todo el datagrama.

IKE I

En los protocolos anteriores es necesario que las dos entidades participantes intercambien una clave común, ya sea para cifrado o autenticación. Dicha clave se obtiene mediante Internet Key Exchange. Se realiza mediante el ISAKMP (Internet Security Association Key Management Protocol), que actúa en dos etapas.

- Las dos entidades negocian los atributos del llamado ISAKMP SA, que incluyen: algoritmo de cifrado, función hash, método de autenticación, parámetros para el protocolo DH y un generador de bits pseudoaleatorio.
- Los atributos anteriores son empleados por una SA para intercambio de claves.

Las dos entidades reciben el nombre de `Initiator` y `Responder`.

IKE II

Las claves de sesión necesarias para establecer una ISAKMP SA dependen del método de autenticación empleado:

$$\texttt{SKEYID} = \text{PBG}(N_i|N_r, g^{x_i x_r}) \text{ para firmas,}$$
$$\texttt{SKEYID} = \text{PBG}(\text{H}(N_i|N_r), CKY_i|CKY_r) \text{ para cifrado de clave pública,}$$
$$\texttt{SKEYID} = \text{PBG}(\texttt{key}, N_i|N_r) \text{ para claves precompartidas,}$$

donde

- N_i, N_r son la carga de datagramas de un solo uso generados por `Initiator` y `Responder`,
- g^{x_i}, g^{x_r} son las claves públicas de `Initiator` y `Responder`, y $g^{x_i x_r}$ la clave DH compartida,
- CKY_i, CKY_r son fichas de `Initiator` y `Responder`, que contienen la identificación de la dirección de las dos entidades,
- PBG es un generador de bits pseudoaleatorio,
- H una función Hash.

IKE III

A partir de la cadena SKEYID se generan tres variantes:

$$\begin{aligned}
\texttt{SKEYID}_d &= \text{PBG}(\texttt{SKEYID}, g^{x_i x_r} | CKY_i | CKY_r | 0),\\
\texttt{SKEYID}_a &= \text{PBG}(\texttt{SKEYID}, \texttt{SKEYID}_d | g^{x_i x_r} | CKY_i | CKY_r | 1),\\
\texttt{SKEYID}_e &= \text{PBG}(\texttt{SKEYID}, \texttt{SKEYID}_a | g^{x_i x_r} | CKY_i | CKY_r | 1)
\end{aligned}$$

empleadas para derivar claves de sesión en SAs, autenticación y confidencialidad. La información intercambiada se autentica mediante las cadenas

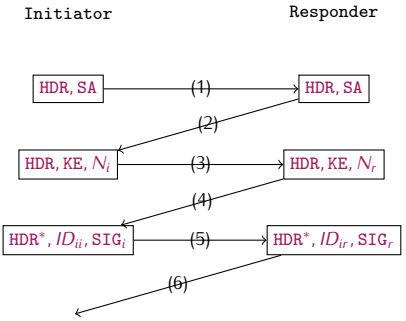
$$\begin{aligned}
H_i &= \text{PBG}(\texttt{SKEYID}, g^{x_i} | g^{x_r} | CKY_i | CKY_r | SA_i | ID_{ii}),\\
H_r &= \text{PBG}(\texttt{SKEYID}, g^{x_r} | g^{x_i} | CKY_r | CKY_i | SA_i | ID_{ir}),
\end{aligned}$$

donde

- SA_i es el cuerpo completo de la carga (payload) menos la cabecera ISAKMP,
- ID_{ii} e ID_{ir} son las cargas identificadoras de `Initiator` y `Responder`.

IKE IV

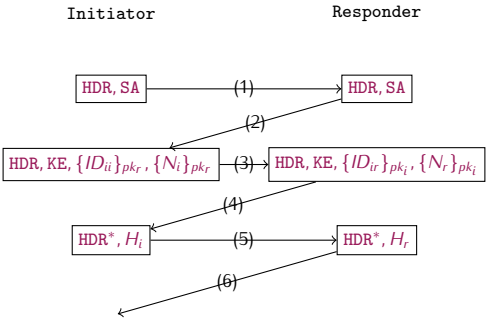
IKE: fase de negociación con firmas



- HDR y HDR* son cabeceras, la segunda con carga cifrada detrás,
- SA, carga de negociación, múltiples opciones en el paso (1) y una sola opción en el paso (2),
- ID_{ii}, ID_{ir}, cargas con identificación,
- SIG$_i$ y SIG$_r$ las firmas de H_i y H_r.

IKE V

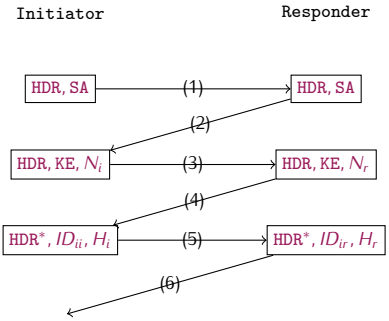
IKE: fase de negociación con cifrado de clave pública



- pk_i y pk_r las claves públicas correspondientes,
- $\{m\}_{pk}$ el mensaje m cifrado con pk.

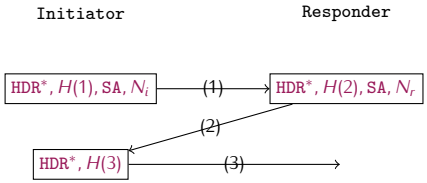
IKE VI

IKE: fase de negociación con clave precompartida



IKE VII

IKE: fase final



- $H(1) = \mathrm{PBG}(\mathtt{SKEYID}_a, M_{ID}|\mathtt{SA}|N_i)$
- $H(2) = \mathrm{PBG}(\mathtt{SKEYID}_a, M_{ID}|\mathtt{SA}|N_r)$
- $H(3) = \mathrm{PBG}(\mathtt{SKEYID}_a, 0|M_{ID}|\mathtt{SA}|N_i|N_r)$
- M_{ID} es el identificador de mensaje de la cabecera del ISAKMP.

Transport Layer Security (TLS) I

El principal objetivo del protocolo TLS es proporcionar privacidad e integridad entre dos aplicaciones en comunicación. Se compone de dos capas: el TLS Record Protocol y el TLS Handshake Protocol, actuando el primero sobre la capa de transporte (TCP).
TLS es independiente de las aplicaciones, pero no indica cómo puede ser usado por ellas.

TLS Record Protocol

Se emplea para encapsular los datos obtenidos de protocolos de más alto nivel.

- La conexión es privada mediante el uso de criptografía simétrica. Las claves son únicas para cada conexión y se obtienen a partir de un secreto negociado por otro protocolo.
- La autenticidad se garantiza mediante el uso de MAC basados en funciones hash.

Transport Layer Security (TLS) II

TLS Handshaking Protocols

Autentica mutuamente a cliente y servidor, y negocia los algoritmos y claves empleados antes del envío de datos.

- La identidad se garantiza mediante criptografía asimétrica. Puede ser opcional, pero normalmente se requiere para al menos uno de los integrantes de la comunicación.
- La negociación del secreto compartido es segura.
- La negociación es confiable. Intentos de modificar la negociación de la comunicación será detectada.

TLS Record Protocol I

Este protocolo funciona mediante la agregación de capas. Cada capa contiene campos con longitudes, descripción y contenidos. El Record Protocol fracciona los datos recibidos de capas superiores, los comprime, aplica un MAC, los cifra y transmite. La parte receptora los descifra, verifica, descomprime y pega.

Estados de conexión

Un estado de conexión es un entorno de operación del TLS Record Protocol. Debe especificar los algoritmos de compresión, MAC y cifrado, así como los parámetros y claves necesarios para su aplicación. Hay cuatro estados presentes: los actuales y los pendientes de lectura y escritura. Todos los registros se procesan mediante los estados actuales. Los parámetros de los estados pendientes se definen en el TLS Handshake Protocol, y pueden convertirse en actuales mediante `ChangeCipherSpec`.

TLS Record Protocol II

Los parámetros de seguridad se establecen mediante los valores siguientes:

- `connection end`, indica si esta entidad es cliente o servidor,
- `PRF algorithm`, función pseudoaleatoria para generar claves a partir del `master secret`,
- `bulk encryption algorithm`, indicando los parámetros necesarios,
- `MAC algorithm`, incluyendo el tamaño de la salida,
- `compression algorithm`,
- `master secret`, un secreto de 48 bytes compartido por ambas entidades,
- `client random`, un valor de 32 bytes proporcionado por el cliente,
- `server random`, un valor de 32 bytes proporcionado por el servidor,

a partir de los cuales se generan los valores

```
client write MAC key
server write MAC key
client write encryption key
server write encryption key
client write IV
server write IV
```

TLS Record Protocol III

Los estados de conexión se instancian convirtiéndolos en estados actuales. Cada estado de conexión contiene los siguientes elementos:

- `compression state`,
- `cipher state`,
- `MAC key`,
- `sequence number`, cada estado de conexión contiene un número de secuencia, mantenido separadamente por los estados de lectura y escritura, se pone a cero cuando un estado se convierte en el estado activo, y no debe exceder $2^{64} - 1$.

TLS Handshaking Protocols I

TLS tiene tres subprotocolos usados para que las entidades integrantes de la comunicación se pongan de acuerdo en los parámetros de seguridad, se autentiquen mutuamente, instancien los parámetros de seguridad negociados, e informen de errores uno a otro.

Una sesión consiste en los siguientes elementos:

- `session identifier`,
- `peer certificate`, de tipo X509v3,
- `compression method`,
- `cipher spec`,
- `master secret`,
- `is resumable`.

TLS Handshaking Protocols II

Change Cipher Spec Protocol

Tanto el cliente como el servidor envían un mensaje, que indica que los siguientes mensajes serán procesados mediante el estado de conexión pendiente.

TLS Handshaking Protocols III

Alert Protocol

Los mensajes de alerta incluyen el nivel de severidad de los mismos (warning, fatal) y la descripción de la alerta:

```
close_notify(0),
unexpected_message(10),
bad_record_mac(20),
decryption_failed_RESERVED(21),
record_overflow(22),
decompression_failure(30),
handshake_failure(40),
no_certificate_RESERVED(41),
bad_certificate(42),
unsupported_certificate(43),
certificate_revoked(44),
certificate_expired(45),
certificate_unknown(46),
illegal_parameter(47),
unknown_ca(48),
access_denied(49),
decode_error(50),
decrypt_error(51),
export_restriction_RESERVED(60),
protocol_version(70),
insufficient_security(71),
internal_error(80),
user_canceled(90),
no_renegotiation(100),
unsupported_extension(110),
(255)
```

TLS Handshaking Protocols IV

Handshake Protocol

Los parámetros criptográficos de una sesión se producen por el `TLS Handshake Protocol`. Está compuesto de los siguientes pasos:

- Intercambiar `hello messages` para acordar algoritmos, intercambiar valores aleatorios y comprobar el reinicio de sesión.
- Intercambiar los parámetros criptográficos que permitan a cliente y servidor acordar un `premaster secret`.
- Intercambiar certificados e información criptográfica que permita a cliente y servidor autenticarse.
- Generar el `master secret` a partir del `premaster secret` y los valores aleatorios intercambiados.
- Proporcionar los parámetros de seguridad al protocolo TLS Record.
- Permitir a cliente y servidor que verifiquen que la otra parte ha calculado los mismos parámetros de seguridad y que no ha habido manipulación por parte de un atacante.

TLS Handshaking Protocols V

Ataques de hombre de en medio pueden tratar de lograr que las dos entidades utilicen métodos lo menos seguros posible. La regla general consiste en que una aplicación no debe permitir la transmisión si el canal de comunicación es menos seguro de lo requerido.

TLS Handshaking Protocols VI

Flujo de un Handshake completo

Cliente		Servidor
ClientHello	$\longrightarrow$	
		ServerHello
		Certificate*
		ServerKeyExchange*
		CertificateRequest*
	$\longleftarrow$	ServerHelloDone
Certificate*		
ClientKeyExchange		
CertificateVerify*		
[ChangeCipherSpec]		
Finished	$\longrightarrow$	
		[ChangeCipherSpec]
	$\longleftarrow$	Finished
Application Data	$\longleftrightarrow$	Application Data

TLS Handshaking Protocols VII

- `ClientHello` y `ServerHello` establecen los atributos `Protocol Version`, `Session ID`, `Cipher Suite`, `Compression Method`, junto con `ClientHello.random`, `ServerHello.random`
- El intercambio de claves se realiza en los mensajes `Certificate`, `ServerKeyExchange` y `ClientKeyExchange`.
- `ServerKeyExchange` se envía si es requerido, ya sea porque el servidor no tiene certificado o porque dicho certificado sólo es válido para firmar.
- Si se ha producido un `CertificateRequest`, el cliente debe enviar su certificado.
- Si el certificado enviado por el cliente permite firmar, el mensaje `CertificateVerify` se envía firmado para asegurar el conocimiento de la clave privada asociada al certificado.

TLS Handshaking Protocols VIII

Flujo de un Handshake abreviado

Cliente		Servidor
ClientHello	$\longrightarrow$	
		ServerHello
		[ChangeCipherSpec]
	$\longleftarrow$	Finished
[ChangeCipherSpec]		
Finished	$\longrightarrow$	
Application Data	$\longleftrightarrow$	Application Data

TLS Handshaking Protocols IX

- El método abreviado se emplea para retomar una sesión anterior o duplicar una sesión existente sin necesidad de negociar nuevos parámetros.
- En el mensaje `ClientHello`, se envía un `Session ID`. Si el servidor encuentra parámetros asociados a este identificador, reinicia la conexión en el estado especificado, enviando su correspondiente `ServerHello` con el mismo `Session ID`. En caso contrario el servidor crea un nuevo `Session ID` y se realiza un Handshake completo.

Índice

Índice

Introducción

- El comercio en Internet ha llegado a depender casi exclusivamente de las instituciones financieras como terceros de confianza en el proceso de los pagos electrónicos.
- Las instituciones financieras no pueden evitar mediar en las disputas.
- El coste de esta mediación incrementa los costes de transacción, limitando su tamaño mínimo útil y eliminando la posibilidad de realizar pequeñas transacciones ocasionales, y hay un coste mayor al perderse la posibilidad de hacer transacciones irreversibles para servicios irreversibles.
- Se acepta como inevitable un cierto porcentaje de fraude.
- Esos costes y la incertidumbre en los pagos se pueden evitar cuando se usa dinero físico en persona, pero no existe mecanismo que permita realizar pagos a través de un canal de comunicación sin la participación de un tercero de confianza.
- Es necesario, por tanto, un sistema de pago electrónico basado en prueba criptográfica en lugar de confianza.
- Si las transacciones son computacionalmente imposibles de revertir, protegerán a los vendedores del fraude, y cualquier mecanismo de depósito de garantía se puede implementar fácilmente para proteger al comprador.

Herramientas: Funciones hash I

Necesitamos una función Hash H de n bits que tenga las propiedades:

- Dado $H(\texttt{nonce}||\texttt{msg})$ es computacionalmente imposible encontrar $\texttt{msg}$.
- Es computacionalmente imposible encontrar dos parejas $(\texttt{nonce}||\texttt{msg})$ y $(\texttt{nonce}'||\texttt{msg}')$ tales que $\texttt{msg} \neq \texttt{msg}'$ y $H(\texttt{nonce}||\texttt{msg}) = H(\texttt{nonce}'||\texttt{msg}')$.
- Para cualquier posible salida $y$de n bits, si elegimos k conteniendo una cadena dentro de una distribución uniforme de n bits, es computacionalmente imposible encontrar x tal que $H(k||x) = y$ en tiempo significativamente menor que 2^n.

Herramientas: Funciones hash II

Un puzle de búsqueda consiste en

- una función Hash H de n bits,
- un valor $\mathtt{id}$ conteniendo una cadena dentro de una distribución uniforme de n bits,
- un conjunto objetivo $Y \subseteq \mathbb{B}^n$.

Una solución del puzle es un valor x tal que

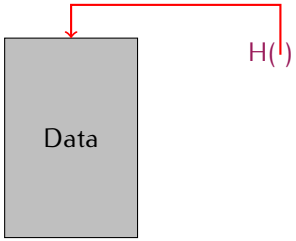
$$\mathrm{H}(\mathtt{id}||x) \in Y.$$

Bitcoin utiliza como función Hash `SHA256` basada en la construcción de Merkle-Damgard.

Herramientas: Blockchain I

Un puntero Hash

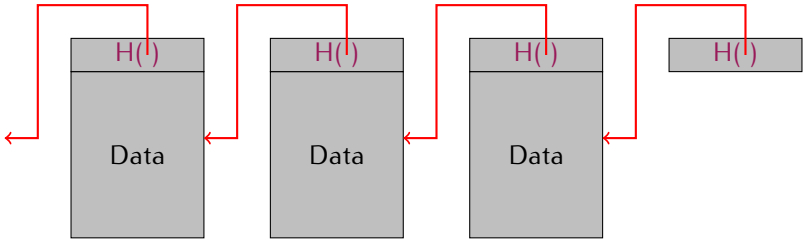
Es un puntero a unos datos junto con el Hash de esos datos.



Herramientas: Blockchain II

Blockchain

Es una lista enlazada mediante punteros Hash.

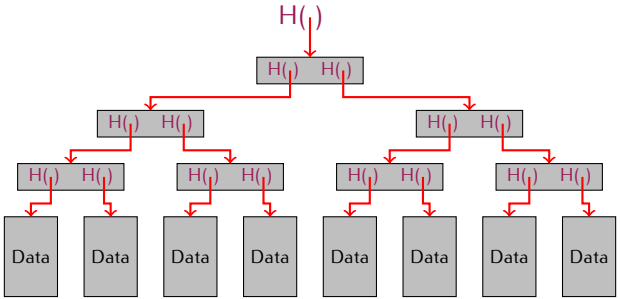


Utilidades

Registro inviolable: Alterar los datos en un bloque requiere recalcular los valores Hash de toda la cadena a partir de ese bloque, incluyendo el último puntero Hash que debe mantenerse guardado.

Herramientas: Árbol de Merkle I

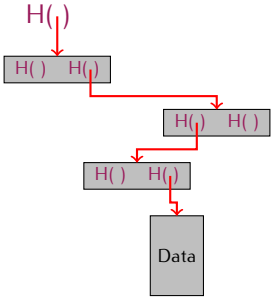
Supongamos que disponemos de un cierto número de bloques que contienen datos. Estos bloques serán las hojas de nuestro árbol.
Agrupamos las hojas en parejas de dos, y para cada pareja construimos una estructura de datos consistente en dos punteros hash, uno a cada uno de los bloques. Repetimos la operación hasta alcanzar un único puntero Hash, la raíz del árbol.



Herramientas: Árbol de Merkle II

Prueba de pertenencia

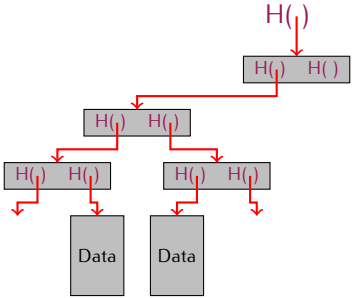
Para certificar que un bloque está en un árbol de Merkle, es suficiente con proporcionar la cadena de punteros Hash que conduce desde la raíz hasta el bloque en cuestión.



Herramientas: Árbol de Merkle III

Prueba de no pertenencia

Si los bloques de datos (las hojas) están ordenados, podemos probar que un bloque no está en el listado dando los caminos al anterior y al posterior.



Herramientas: ECDSA

secp256k1

```
Field Type: prime-field
Prime:
  00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
  ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:fe:ff:
  ff:fc:2f
A: 0
B: 7 (0x7)
Generator (uncompressed):
  04:79:be:66:7e:f9:dc:bb:ac:55:a0:62:95:ce:87:
  0b:07:02:9b:fc:db:2d:ce:28:d9:59:f2:81:5b:16:
  f8:17:98:48:3a:da:77:26:a3:c4:65:5d:a4:fb:fc:
  0e:11:08:a8:fd:17:b4:48:a6:85:54:19:9c:47:d0:
  8f:fb:10:d4:b8
Order:
  00:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:ff:
  ff:fe:ba:ae:dc:e6:af:48:a0:3b:bf:d2:5e:8c:d0:
  36:41:41
Cofactor: 1 (0x1)
```

Consenso distribuido

Protocolo

Partimos de N nodos, cada uno con un valor de entrada. Algunos de esos nodos son maliciosos o descuidados. Un protocolo de consenso distribuido tiene las siguientes propiedades:

- debe finalizar con todos los nodos honestos de acuerdo en un valor común,
- el valor debe haber sido generado por un nodo honesto.

Algoritmo de consenso de Bitcoin (simplificado)

La simplificación se encuentra en que se asume la posibilidad de seleccionar un nodo arbitrario de forma no vulnerable al ataque de Sybil.

1. Las nuevas transacciones se envían a todos los nodos.
2. Cada nodo recolecta las nuevas transacciones en un nodo.
3. En cada ronda un nodo consigue transmitir a los demás su bloque.
4. Los demás nodos aceptan el bloque solo si todas las transacciones son válidas.
5. Los nodos aceptan el bloque incluyendo su hash en el siguiente bloque que generan

Transacciones I

- Una moneda electrónica es una cadena de firmas digitales.
- Cada propietario transfiere la moneda al siguiente propietario firmando digitalmente un hash de la transacción previa y la clave pública del siguiente propietario, y añadiendo ambos al final de la moneda.
- El beneficiario puede verificar las firmas para verificar la cadena de propiedad.
- El beneficiario no puede verificar que uno de los propietarios no haya gastado dos veces la misma moneda.
- La solución habitual es introducir una autoridad central de confianza, o casa de la moneda, que comprueba cada transacción para que eso no se produzca.
- Tras cada transacción, la moneda debe regresar a la casa de la moneda para distribuir una nueva moneda, y solo las monedas emitidas directamente desde ella están libres de la sospecha de doble gasto.
- El destino de todo el sistema de dinero depende de la compañía que gestiona la casa de la moneda, por la cual pasa cada transacción.

Transacciones II

- Necesitamos una forma de que el beneficiario sepa que los propietarios previos no han firmado transacciones anteriores. La única manera de confirmar la ausencia de una transacción es tener conocimiento de todas las transacciones. En el modelo de la casa de la moneda, ésta tiene conocimiento de todas las transacciones y decide cuáles llegaron primero.
- Para lograr esto sin la participación de una parte de confianza, las transacciones han de ser anunciadas públicamente, y necesitamos un sistema para que los participantes estén de acuerdo en un único historial del orden en que fueron recibidas.
- El beneficiario necesita prueba de que en el momento de la transacción la mayor parte de los nodos estaban de acuerdo en que esa fue la primera que se recibió.

Sellado de tiempo

- Un servidor de sellado de tiempo trabaja tomando el hash de un bloque de ítems para sellarlos en el tiempo y notificar públicamente su *hash*.
- El sellado de tiempo prueba que los datos han existido en el tiempo, obviamente, para poder entrar en el hash.
- Cada sellado de tiempo incluye el sellado de tiempo previo en su hash, formando una cadena, con cada sellado de tiempo adicional reforzando al que estaba antes.

Proof-of-work I

- La proof-of-work consiste en buscar un valor cuyo hash comience con un número de bits cero.
- El trabajo medio que hace falta es exponencial en el número de cero bits requeridos y puede verificarse ejecutando un único hash.
- Para nuestra red de sellado de tiempo, implementamos la proof-of-work incrementando un nonce en el bloque hasta que se encuentre un valor que otorgue al hash del bloque los cero bits requeridos.
- Una vez que se ha agotado el esfuerzo de CPU para satisfacer la proof-of-work, el bloque no se puede cambiar sin rehacer el trabajo.
- A medida que bloques posteriores se encadenen tras él, el trabajo para cambiar un bloque incluiría rehacer todos los bloques anteriores.
- La proof-of-work también resuelve el problema de determinar la representación en la toma de decisiones mayoritarias.
- La proof-of-work es esencialmente un voto por CPU.
- La decisión de la mayoría está representada por la cadena más larga, en la cual se ha invertido el mayor esfuerzo de proof-of-work.

Proof-of-work II

- Si la mayoría de la potencia CPU está controlada por nodos honestos, la cadena honesta crecerá más rápido y dejará atrás cualquier cadena que compita.
- Para modificar un bloque pasado, un atacante tendría que rehacer la proof-of-work del bloque y de todos los bloques posteriores, y entonces alcanzar el trabajo de los nodos honestos.
- Para compensar el aumento en la velocidad del hardware y el interés variable de los nodos activos a lo largo del tiempo, la dificultad de la proof-of-work está determinada por una media móvil que apunta a un número medio de bloques por hora. Si se generan muy deprisa, la dificultad aumenta.

Red I

- Los pasos para ejecutar la red son:
  1. Las transacciones nuevas se transmiten a todos los nodos.
  2. Cada nodo recoge todas las transacciones en un bloque.
  3. Cada nodo trabaja en resolver una proof-of-work compleja para su bloque.
  4. Cuando un nodo resuelve una proof-of-work, transmite el bloque a todos los nodos.
  5. Los nodos aceptan el bloque si todas las transacciones en él son válidas y no se han gastado con anterioridad.
  6. Los nodos expresan su aceptación del bloque al trabajar en crear el siguiente bloque en la cadena, usando el hash del bloque aceptado como hash previo.
- Los nodos siempre consideran correcta a la cadena más larga y se mantendrán trabajando para extenderla. Si dos nodos transmiten simultáneamente diferentes versiones del siguiente bloque, algunos nodos recibirán una antes que la otra. En ese caso, trabajarán sobre la primera que hayan recibido, pero guardarán la otra ramificación por si acaso se convierte en la más larga.
- El empate se romperá cuando se encuentre la siguiente proof-of-work y una ramificación se convierta en la más larga; los nodos que trabajaban en la otra ramificación cambiarán automáticamente a la más larga.

Red II

- La transmisión de nuevas transacciones no precisa alcanzar todos los nodos, con alcanzar a la mayoría de los nodos, entrarán en un bloque en poco tiempo.
- Las transmisiones de nodos también toleran mensajes perdidos. Si un nodo no recibe un bloque, lo reclamará cuando reciba el siguiente bloque y se dé cuenta de que falta uno.

Incentivo I

- Por convenio, la primera transacción en un bloque es una transacción especial con la que comienza una moneda nueva, propiedad del creador del bloque.
- Esto añade un incentivo a los nodos para soportar la red, y proporciona una forma de poner las monedas en circulación, dado que no hay autoridad central que las distribuya.
- La inserción estable de una cantidad constante de monedas nuevas es análoga at trabajo de mineros de oro, que consumen recursos para añadir oro a la circulación. En nuestro caso, es tiempo de CPU y electricidad lo que se gasta.
- El incentivo también se basa en las comisiones por transacción. Si el valor de salida de una transacción es menor que el valor de entrada, la diferencia es una comisión por transacción que se añade al valor de incentivo del bloque que contiene la transacción. Una vez que un número predeterminado de monedas ha entrado en circulación, el incentivo puede evolucionar hacia comisiones de transacción y estar completamente libre de inflación.

Incentivo II

- El incentivo puede ayudar a que los nodos permanezcan honestos. Si un atacante codicioso fuera capaz de reunir más potencia CPU que la de todos los nodos honestos, tendría que escoger entre usarla para defraudar a la gente robándoles los pagos recibidos, o usarla para generar nuevas monedas. Debe encontrar más rentable respetar las reglas, esas reglas que le favorecen entregándole más monedas nuevas que a todos los demás en conjunto, que socavar el sistema y la validez de su propia riqueza.

Espacio de disco

- Cuando la última transacción de una moneda está enterrada bajo suficientes bloques, las transacciones que se han gastado antes que ella se pueden descartar para ahorrar espacio de disco.
- Para facilitar esto sin romper el hash del bloque, los valores hash de las transacciones almacenados en un Árbol de Merkle, incluyendo solo la raíz en el hash del bloque. Los bloques viejos pueden compactarse podando ramas del árbol. Los hashes interiores no necesitan ser guardados.
- Una cabecera de bloque sin transacciones pesaría unos $80B$. Si suponemos que los bloques se generan cada 10 minutos, $80B \times 6 \times 24 \times 365 = 4{,}2MB$ por año. Siendo habitual la venta de ordenadores con 2GB de RAM en 2008, y con la Ley de Moore prediciendo un crecimiento de $1{,}2GB$ anual, el almacenamiento no debería suponer un problema incluso si hubiera que conservar en la memoria las cabeceras de bloque.

Verificación de pagos simplificada I

- Es posible verificar pagos sin ejecutar un nodo plenamente en red. El usuario solo necesita tener una copia de las cabeceras de bloque de la cadena más larga de proof-of-work, que puede conseguir solicitándola a los nodos de red hasta estar convencido de que tiene la cadena más larga, y obtener la rama Merkle que enlaza la transacción con el bloque en que está sellado en el tiempo.
- El usuario no puede comprobar la transacción por sí mismo pero, al enlazarla a un lugar en la cadena, puede ver que un nodo de la red la ha aceptado, y los bloques añadidos posteriormente confirman además que la red la ha aceptado.
- la verificación es fiable en tanto que los nodos honestos controlen la red, pero es más vulnerable si un atacante domina la red.
- Mientras que los nodos de red pueden verificar las transacciones por sí mismos, el método simplificado puede ser engañado por transacciones fabricadas por un atacante, en tanto el atacante pueda continuar dominando la red.

Verificación de pagos simplificada II

- Una estrategia para protegerse contra esto podría ser aceptar alertas de los nodos de red cuando detecten un bloque no válido, sugiriendo al software del usuario que descargue el bloque entero y las transacciones con aviso para confirmar la inconsistencia.
- Los negocios que reciban pagos con frecuencia seguramente preferirán tener sus propios nodos ejecutándose para tener más seguridad independiente y verificación más rápida.

Combinando y dividiendo valor

- Aunque sería posible manipular monedas individualmente, no sería manejable hacer una transacción para cada céntimo en una transferencia.
- Para permitir que el valor se divida y combine, las transacciones contienen múltiples entradas y salidas.
- Normalmente habrá, o bien una entrada simple de una transacción anterior mayor, o bien múltiples entradas combinando pequeñas cantidades, y como máximo dos salidas: una para el pago y otra devolviendo el cambio, si lo hubiera, al emisor.
- Cabe señalar que la diseminación de control, donde una transacción depende de varias transacciones, y esas transacciones dependen de muchas más, no supone aquí un problema. No existe la necesidad de extraer una copia completa e independiente del historial de una transacción.

Privacidad

- El modelo tradicional de banca consigue un nivel de privacidad limitando el acceso a la información a las partes implicadas y el tercero de confianza.
- La necesidad de anunciar públicamente todas las transacciones excluye este método, pero aún así se puede mantener la privacidad rompiendo el flujo de información en otro punto: manteniendo las claves públicas anónimas.
- El público puede ver que alguien está enviando una cantidad a otro alguien, pero sin que haya información vinculando la transacción con nadie. Esto es similar al nivel de información que comunican las bolsas de valores, donde el tiempo y tamaño de las operaciones individuales, la "cinta", son hechos públicos, pero sin decir quiénes fueron las partes.
- Como cortafuegos adicional, debería usarse un nuevo par de claves en cada transacción para evitar que se relacionen con un propietario común.
- Con las transacciones multientrada será inevitable algún tipo de vinculación, pues revelan necesariamente que sus entradas pertenecieron al mismo propietario.
- El riesgo es que si se revela la identidad del propietario de una clave, la vinculación podría revelar otras transacciones que pertenecieron al mismo propietario.

Por qué funciona. I

- Consideramos el escenario de un atacante intentando generar una cadena alternativa más rápido que la cadena honesta.
- Incluso si se consigue, el sistema no queda abierto a cambios arbitrarios como crear valor de la nada o tomar dinero que nunca perteneció al atacante. Los nodos no van a aceptar una transacción inválida como pago y los nodos honestos nunca aceptarán un bloque que las contenga.
- Un atacante solo puede tratar de cambiar una de sus propias transacciones para recuperar dinero que ha gastado recientemente.
- La carrera entre la cadena honesta y la cadena de un atacante puede verse como un camino aleatorio binomial. El suceso que prospera es la cadena honesta añadiendo un bloque, aumentando su liderato por $+1$, y el suceso que fracasa es la cadena del atacante añadiendo un bloque, reduciendo la brecha en -1.
- La probabilidad de que un ataque alcance la cadena honesta desde un déficit dado es análoga al problema de la ruina del jugador.

Por qué funciona. II

- Supongamos que un jugador con crédito ilimitado comienza con un déficit y juega en potencia un número infinito de intentos para alcanzar un punto de equilibrio. Podemos calcular la probabilidad de que alcance ese punto, o de que un ataque alcance alguna vez a la cadena honesta

$$
\begin{aligned}
p &= \text{probabilidad de que un nodo honesto encuentre el siguiente bloque} \\
q &= \text{probabilidad de que el atacante encuentre el siguiente bloque} \\
q_z &= \text{probabilidad de que el atacante alcance la cadena honesta desde } z \text{ bloques atrás}
\end{aligned}
$$

$$
q_z = \begin{cases} 1 & \text{si } p \leq q \\ (q/p)^z & \text{si } p > q \end{cases}
$$

- Asumiendo que $p > q$, la probabilidad cae de forma exponencial a medida que aumenta el número de bloques que el atacante tiene que alcanzar. Con las probabilidades en su contra, si no tiene un golpe de suerte que lo haga avanzar desde el principio, sus oportunidades se irán desvaneciendo a medida que se va quedando atrás.

Por qué funciona. III

- Cuánto tiempo necesita esperar el receptor de una transacción para tener la suficiente seguridad de que el emisor no puede cambiarla. Asumimos que el emisor es un atacante que quiere que el receptor crea durante un tiempo que le ha pagado; entonces cambiará el pago para devolvérselo a sí mismo un tiempo después. El receptor recibirá un aviso cuando esto suceda, pero el emisor espera que ya sea demasiado tarde.
- El receptor genera un nuevo par de claves y da la clave pública al emisor poco antes de firmar. Esto impide que el emisor pueda preparar una cadena de bloques previa trabajando de continuo en ella hasta tener la suerte suficiente como para ponerse a la cabeza y, entonces, ejecutar la transacción. Una vez que la transacción se ha emitido, el emisor deshonesto comienza a trabajar en secreto en una cadena paralela que contiene una versión alternativa de su transacción.
- El receptor espera hasta que la transacción se ha añadido al bloque y z bloques se han enlazado tras él. No sabe la cantidad de progreso que ha realizado el atacante, pero asumiendo que los bloques honestos han tomado la media de tiempo esperada por bloque, el potencial de progreso del atacante será una distribución de Poisson9 con un valor esperado $\lambda = z\frac{q}{p}$.

Por qué funciona. IV

- Para obtener la probabilidad de que el atacante pueda ponerse al día incluso ahora, multiplicamos la densidad de Poisson para cada aumento en el progreso que podría haber realizado, por la probabilidad que podría haber alcanzado desde ese punto:

$$\sum_{k\geq 0} \frac{\lambda^k e^{-k}}{k!} \cdot \begin{cases} (q/p)^{z-k} & \text{si } k \leq z \\ 1 & \text{si } k > z \end{cases} = 1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-k}}{k!} \left(1 - (q/p)^{z-k}\right)$$

Índice

Índice

Identidad I

- Una identidad digital es la información usada para representar una entidad en un sistema de tecnologías de la información y la comunicación (ICT). El propósito del sistema ICT determina qué atributos de los que describen una entidad son usados para una identidad. Dentro del un sistema ICT, una identidad está compuesta por los atributos relacionados con una entidad relevantes para el dominio particular de aplicaciones proporcionadas por el sistema ICT. Dependiendo de los requisitos específicos del dominio, el conjunto de atributos relacionados con la entidad (la identidad) puede, pero no tiene por qué, distinguir de forma única a dicha entidad de otras entidades en el sistema ICT.
- Dentro de un mismo sistema ICT, el conjunto de atributos que determinan una identidad puede ser el mismo o no.
- Una misma entidad puede tener diversas identidades, cada una de ellas asociada a un dominio específico. Una entidad puede tener varias identidades incluso en el mismo dominio. Si una identidad no es única en un dominio particular, la identidad puede emplearse para distinguir el grupo de entidades que comparten en común los atributos constituyentes de dicha identidad.

Identidad II

- La información de una entidad en un dominio se llama información identificativa.
- Si dicha información permite diferenciar en el dominio a una entidad de otras, nos referimos a ellas como identidad distinguida.
- Si la combinación de valores contenidos en la información identificativa es única en el dominio, se convierten en un identificador de la entidad.
- Con la elaboración de una nueva identidad de una entidad en un dominio asignamos valores a los atributos requeridos:
  - interacción entre dominio y entidad,
  - identificación futura de la entidad, incluyendo aspectos físicos,
  - autenticación futura de la identidad de la entidad,
  - uno o más identificadores de referencia.
- La información identificativa puede cambiarse. Los medios para cambiar, actualizar o crear información identificativa deben ser especificados, incluyendo en mantenimiento de registros que permitan la auditoría.

Identidad III

- Un identificador puede estar:
  - disponible para el uso exclusivo del dominio de origen,
  - permitido para su uso en otros dominios.
- El objeto físico en el que se almacena un identificador debe estar equipado con medidas de seguridad que garanticen las características usuales: integridad, confidencialidad, autenticidad...

Atributos I

Cada atributo tiene un tipo, valor y contexto operacional. El contexto operacional pued ser el dominio de origen o el dominio de aplicación. Una posible clasificación de tipos es la siguiente:

- información sobre la existencia física de la entidad,
- información sobre la evolución de la entidad en el tiempo,
- información intrínseca a la existencia física de la entidad,
- información asignada a la entidad,
- referencia a un objeto que representa la información identificativa de la entidad.

Gestión de la información identificativa I

La gestión de la identidad incluye la gobernanza, políticas, procesos, datos, tecnología y estándares que pueden incluir:

- Aplicaciones que implementen el registro de una identidad.
- Autenticación de la identidad.
- Establecimiento de la procedencia de la información identificativa.
- Establecimiento de la conexión entre la entidad y la información identificativa.
- Mantenimiento de la información identificativa.
- Garantía de la integridad de la información identificativa.
- Credenciales y servicios que faciliten la autenticación de la entidad como identidad conocida.
- Minimización del riesgo de robo o uso inapropiado de la información.

Gestión de la información identificativa II

El ciclo de vida de una identidad incluye las siguientes etapas:

- **desconocido** no existe información que pueda ser empleada para identificar a la entidad,
- **establecido** la información requerida ha sido verificada, información adicional ha sido generada y todo ha sido correctamente almacenado.
- **activo** la información identificativa está presente en el sistema de gestión de la identidad, lo que permite la interacción y uso de los recursos por parte de la entidad.
- **suspendido** la información identificativa permite al sistema de gestión de la identidad denegar el uso de los recursos del dominio a la entidad
- **arvhivado** el registro de la información identificativa se mantiene incluso más allá de la existencia de la entidad en el dominio.

Gestión de la información identificativa III

Existen las siguientes transiciones entre la etapas anteriores

inscripción incluyendo prueba de identidad y el registro de una entidad, incluyendo información identificativa generada y verificada.

activación se agrega la información identificativa al registro para que la entidad pueda interactuar con los servicios proporcionados en el dominio.

mantenimiento actualización de la información identificativa en el registro.

ajuste de identidad actualización que implica cambio en la información de activación

suspensión parte de la información identificativa de una entidad se marca como no disponible temporalmente.

reactivación proceso inverso del anterior.

borrado de la información identificativa almacenada.

archivo borrado parcial de la información identificativa, de forma que ésta permanece almacenada con fines estadísticos.

reestablecimiento es un proceso de inscripción en el que parte de la información identificativa se obtiene de información previamente almacenada.

Identificación I

El proceso de identificación destablece

- que la entidad es conocida por el dominio,
- que la entidad es acreditada para ser considerada como conocida por el dominio.

La identificación utiliza la información identificativa para determinar si

- la identidad existe para la entidad,
- la entidad coincide con la información identificativa conocida, presentada u observada,
- la entidad está únicamente asociada con la identidad.

Identificación II

Los atributos asociados por la identificación a la identidad y la entidad pueden ser

- determinados por la observación,
- proporcionados por la entidad,
- recuperados por el registro de identidad,
- proporcionados por otra fuente,
- asignados durante el proceso.

Identificación III

Verificación

Se debe verificar la nueva información identificativa, la proporcionada por un registro de identidad u otro proveedor. Se debe garantizar que la información identificativa

- está presente en un formato aprobado,
- contiene un valor admisible con los criterios específicos del dominio y el propósito de la identificación,
- ha sido originada dentro del periodo de validez requerido,
- ha sido originada por una fuente fiable.

Identificación IV

Inscripción

- La inscripción puede derivar en la creación de una o varias identidades asociadas a una entidad. En particular se crea un identificador de referencia.
- La información identificativa creada se registra en un dominio como la identidad de la entidad inscrita.
- Evidencias de la identidad pueden ser también registradas en este momento de la inscripción.
- Los atributos que determinan unívocamente a la identidad pueden ser escogidos por la entidad o por el sistema de gestión de identidades.
- Datos biométricos pueden ser incluidos en este momento.

Autenticación

Un sistema de gestión de la identidad debe especificar para cada uno de sus procesos de autenticación

- políticas para la verificación de la información identificativa,
- mecanismos para establecer la validez y la corrección de una identidad autenticada,
- el periodo de validez de una identidad autenticada,
- mecanismos para el registro, auditoría, procesamiento y resultados (incluidos los intermedios).

Mantenimiento

- Un sistema de gestión de la identidad debe realizar el mantenimiento de la información identificativa que tiene registrada cambiando el valor de uno o varios atributos en una identidad.
- Debe, por tanto, especificar los mecanismos para mantener la integridad y la fiabilidad de los atributos que almacena.
- Una autoridad de información identificativa debe proporcionar los datos disponibles más fiables de una identidad, respetando la privacidad.

Aspectos de implementación

Centralizada Un único registro de la identidad, un único punto de control sobre la inscripción y acceso a la información identificativa almacenada.

Distribuida Múltiples registros de la identidad, múltiples puntos de control sobre la inscripción y acceso a la información identificativa almacenada.

Centrada en el usuario Permite a las entidades jugar un papel activo en la gestión de la información identificativa almacenada en el registro de la identidad.

Federada Permite a un sistema de gestión de la identidad que no contiene información identificativa en su propio registro confiar en la información identificativa contenida en otros sistemas de gestión de la identidad, que actúan como autoridad de información identificativa en este caso.

Privacidad

Un sistema de gestión de identidad debe proporcionar capacidades relacionadas con la privacidad:

- Implementar mecanismos, políticas, procesos y tecnología para minimizar revelaciones.
- Autenticar entidades que emplean información identificativa.
- Minimizar la capacidad para enlazar identidades.
- Registrar y auditar el uso de información identificativa.
- Proteger contra riesgos generados inadvertidamente.
- Implementar políticas para la revelación selectiva de datos.
- Permitir el uso de seudónimos.
- Implementar políticas para contratar una entidad humana con el fin de obtener el consentimiento para actividades relacionadas con su información identificativa sensible.

Índice

Concepto

- La gestión de la identidad alude a las políticas, procesos y tecnologías que establecen la información identificativa e impone las reglas sobre el acceso a los recursos digitales.
- Un proceso de autenticación determina a qué sistemas puede acceder una entidad autorizada.
- Dentro de una organización con varios sistemas, en lugar de emplear un sistema de gestión de la identidad para cada uno de ellos se centraliza la gestión de la identidad, y cada sistema autoriza el acceso a sus recursos en función del valor de la información identificativa de la entidad.
- La gestión federada de la identidad extiende esta idea creando una autoridad de confianza para identidades digitales a través de múltiples organizaciones.
- En un sistema federado las organizaciones participantes comparten atributos basándose en estándares consensuados.
- Se permite el acceso de otros miembros de la federación a recursos instalados en un dominio dentro de la misma.

Funcionamiento

- Cuando una entidad pide acceso a un recurso, se le solicita la información identificativa, entre la que se incluye la organización anfitriona de la entidad.
- La solicitud se envía a la sistema de gestión de identidad de la organización anfitriona la cual, una vez comprobados los permisos asociados a la identidad, informa a la organización solicitante de que la entidad ha sido autenticada.
- Las organizaciones federadas deciden qué atributos de las entidades serán compartidos, normalmente nombre, título y posición.
- En base a esta información y sus respectivas políticas, se concede o niega el acceso a recursos concretos.
- La información identificativa de cada usuario permanece en un único sistema.
- La gestión federada de la identidad separa el acceso del establecimiento de la identidad y la autorización.
- Cada institución almacena la información identificativa de sus propios usuarios, y no la de los demás.
- La verificación se realiza en la institución anfitriona, que por otra parte mantiene los datos más actuales y fiables.

Paradigma

- Se simplifica la administración y el flujo en el acceso a los recursos.
- Establecer nuevos servicios es sencillo puesto que el proceso de identificación ya está establecido.
- La gestión federada de la identidad pone el foco en usuarios y servicios en lugar de en la organización.
- Al eliminar la réplica de las bases de datos de información identificativa, la gestión de la identidad mejora la seguridad tanto para las entidades como para los recursos.

Inconvenientes

- Necesaria la modificación de los sistemas y aplicaciones existentes.
- Es necesario uniformizar algunos atributos de la información identificativa entre los organismos federados.
- La participación en varias federaciones puede conllevar diferentes requisitos, no necesariamente compatibles.
- La diferencia de criterios para autenticar el acceso puede conllevar la denegación del uso de usuarios que deberían estar autorizados y el acceso a otros no autorizados.

Ejemplos

- eduroam
- facebook
- google
- roaming
- OpenID
- ORCID

Índice

Sistemas biométricos I

- Un sistema biométrico es un sistema de reconocimiento de patrones que reconoce a una persona a partir de un vector de características derivadas de rasgos fisiológicos o comportamientos específicos que posee.
- Los sistemas biométricos pueden actuar en modo verificador o identificador.
- El modo identificador compara la información biométrica adquirida con toda la base de datos de usuarios.
- El modo verificador compara la información biométrica adquirida con aquella correspondiente a la identidad reclamada.

Un sistema biométrico consta de cuatro componentes:

- Sensor
- Extractor de características
- Emparejamiento
- Decisión

Sistemas biométricos II

Requerimientos

- Universalidad
- Diferencia
- Permanencia
- Cobrabilidad
- Rendimiento
- Aceptabilidad
- Burla

Ejemplos I

Termograma de infrarrojos

- Mide el patrón del calor irradiado por en cuerpo humano.
- No invasivo.
- Adquisición dificultada por fuentes externas de calor.
- Adquisición dificultada por la tridimensionalidad.
- Puede afinarse para medir la disposición de las venas de la mano.

Paso

- Es la forma peculiar en que uno camina, una característica espacio-temporal compleja.
- No es especialmente diferente, puede emplearse como medida de seguridad baja.
- Tampoco es permanente en el tiempo.
- Al necesitar secuencias de vídeo, es computacionalmente costoso.

Ejemplos II

Pulsación del teclado

- Se cree que cada persona pulsa el teclado de forma característica.
- No es muy diferente, pero puede emplearse para verificación.
- Es especialmente no intrusivo.

Olor

- Es característico de los compuestos químicos esparcidos por cada objeto, lo que puede emplearse para distinguirlos.
- Los perfumes y desodorantes pueden disminuir la diferencia entre entidades.

Oreja

- Se ha sugerido que la estructura cartilaginosa del pabellón de la oreja es distintiva, aunque no hay consenso en ello.
- Empareja la distancia entre salientes del pabellón dentro de un punto de referencia de la oreja.

Ejemplos III

Geometría de la mano

- Se compara la longitud de los dedos, la localización de las articulaciones, y la forma y tamaño de la palma.
- Es antiguo, en uso desde el final de los 60.
- No parece verse afectado por situaciones de humedad o temperatura.
- Al no ser muy diferente puede usarse para verificación, pero no para identificación.
- Puede no ser permanente (crecimiento, artritis) y se ve afectado por joyas y similares.
- Puede simplificarse midiendo la geometría parcial de la mano.

Ejemplos IV

Huella dactilar

- Mide las crestas y surcos en la yema de cada dedo.
- Es también de los más antiguos, inicialmente se almacenaban en papel mediante tinta.
- Los sensores actuales exploran una imagen de las yemas, lo que deteriora la exploración con el uso por la suciedad y restos que los dedos aportan, o emplean un chip de silicio recubierto que mide la capacidad eléctrica para medir las crestas y formar una imagen.
- Las características extraídas se corresponden con la posición y orientación de ciertos puntos críticos llamados puntos minuciosos.
- Para identificación requiere una gran cantidad de recursos computacionales.

Ejemplos V

Rostro

- Es la característica biométrica más común usada por los seres humanos.
- Puede extraerse de forma estática o dinámica (aeropuertos, metro...)
- Las aproximaciones más populares al reconocimiento facial se basa en dos métodos,
  - localización y forma de atributos faciales como ojos, cejas, nariz, labios y barbilla, y sus relaciones espaciales,
  - un análisis global de la imagen de una cara como una combinación ponderada de caras canónicas.
- Cambios en el ángulo y la iluminación pueden dificultar la tarea de extracción de los patrones.
- Maquillaje y cambios de expresión pueden también alterar la extracción de patrones.

Ejemplos VI

Retina

- La configuración vascular de la retina de cada ojo es una característica única.
- Al estar protegida dentro del ojo y ser, por consiguiente, difícil de replicar, es una de las medidas biométricas más seguras.
- La adquisición de la imagen requiere la cooperación del sujeto.
- La exploración de la retina puede revelar problemas médicos.

Iris

- El iris es un patrón complejo que contiene muchas características diferenciadoras, como ligamentos arqueados, surcos, crestas, criptas, anillos, corona, pecas y un cuellos en zigzag.
- La medición del iris es menos intrusiva que la retina, pues puede tomarse la imagen incluso desde varios metros de distancia.
- Las respuestas del iris a los cambios de luz también verifican que el sujeto está vivo.
- El iris es permanente en adultos, pero presenta variaciones en la adolescencia.

Ejemplos VII

Huella de la palma

- Similar a la huella dactilar pero de toda la palma de la mano.
- Más fiable que la huella pero más costosa de extraer y almacenar.
- Incrementa su fiabilidad con el análisis de la geometría de la mano.

Voz

- Las características individuales de la voz se basan en las vías vocales, boca, cavidades nasales y labios que se utilizan para crear sonidos.
- Tienen un aspecto individual e invariante, pero otro asociado al comportamiento que cambia con el tiempo, situación médica y estado emocional.
- Los sistemas deben ser entrenados en el momento de la inscripción, y suelen necesitar más de una sesión.
- Los algoritmos de extracción de características son similares al reconocimiento facial.

Ejemplos VIII

Firma

- El modo en que un individuo escribe su nombre es una característica individual.
- La recogida de muestras requiere cooperación y útiles de escritura.
- Es una característica del comportamiento y como tal sujeta a cambios a lo largo del tiempo y a las condiciones físicas y emocionales.
- Además de la forma, se mide la presión y velocidad en la escritura.

ADN

- Es la medida biométrica más fiable. Es única excepto en gemelos univitelinos.
- Tiene algunos inconvenientes:
  - contaminación y sensibilidad,
  - a día de hoy no es analizable en tiempo real,
  - la privacidad se puede ver comprometida por poder detectarse problemas médicos.

Rendimiento I

- Es imposible que dos muestras de la misma característica biométrica tomadas en dos sesiones distintas sean iguales.
- La respuesta de un sistema de emparejamiento biométrico suele ser un valor numérico s, llamada puntuación, que mide la similitud entre la entrada y la muestra almacenada.
- La puntuación s se compara con un valor umbral t, si $s \geq t$ se concluye que las muestras pertenecen a la misma persona.
- La distribución de puntuaciones generada por parejas de muestras de personas diferentes se llama *distribución impostora*.
- La distribución de puntuaciones generada por parejas de muestras de la misma persona se llama *distribución genuina*.

Rendimiento II

Errores

Existen dos tasas de error asociadas a los sistemas biométricos.

Falso rechazo (TFR) Dar como distintas muestras de la misma persona.

Falsa aceptación (TFA) Dar como iguales muestras de personas distintas.

Estas tasas dependen fundamentalmente del umbral t. A mayor t, mayor TFR y menor TFA, y viceversa.
Hay otros dos tipos de error, el fallo en la captura de datos y el fallo en la inscripción, que miden la probabilidad de fallo en la obtención de los datos por parte del sensor y la probabilidad de fallo en la finalización correcta del proceso de inscripción respectivamente.

Característica	TFR	TFA
Huella dactilar	0,2 %	0,2 %
Rostro	10 %	1 %
Voz	$10 - 20$ %	$2 - 5$ %

Rendimiento III

Limitaciones

- Ruido en los sensores.
- Variaciones internas. Causadas normalmente por una interacción incorrecta.
- Diferencia. Puede no ser suficientemente destacada como para ser medida.
- No universalidad.
- Suplantación. Sobre todo en firma y voz.

Una solución a estas limitaciones es el diseño de sistemas multimodales que implementas varias características biométricas diferentes en el mismo sistema.

Bibliografía I


C. Adams, P. Cain, D. Pinkas, and R. Zuccherato.
Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP).
IETF, August 2001.


Ingemar J. Cox, Matthew L. Miller, Jeffrey A. Bloom, Jessica Fridrich, and Ton Kalker.
Digital Watermarking and Steganography.
The Morgan Kaufmann Series in Multimedia Information and Systems. Elsevier, second edition edition, 2008.


D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk.
Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.
IETF, May 2008.


Quynh H. Dang.
The Keyed-Hash Message Authentication Code (HMAC).
National Institute of Standards and Technology (NIST), July 2008.

Bibliografía II


Kresimir Delac and Mislav Grgic.
A survey of biometric recognition methods.
In *46th International Symposium Electronics in Marine, ELMAR-2004*, pages 184–193, 2004.


Hans Delfs and Helmut Knebl.
Introduction to Cryptography. Principles and Applications.
Information Security and Cryptography. Springer, third edition, 2015.


T. Dierks and E. Rescorla.
The Transport Layer Security (TLS) Protocol Version 1.2.
IETF, August 2008.


Educause.
7 things you should know about Federared Identity Management, September 2009.

Peter Gutmann.
Everything you Never Wanted to Know about PKI but were Forced to Find Out.
Technical report, University of Auckland.

Bibliografía III


ISO/IEC.
Information technology — Security techniques — A framework for identity management —, December 2011.


Joseph Migga Kizza.
Guide to Computer Network Security.
Computer Communications and Networks. Springer, 3rd. edition, 2015.


S. Kent and K. Seo.
Security Architecture for the Internet Protocol.
IETF, December 2005.


Andre Karamanian, Srinivas Tenneti, and Francois Dessart.
PKI Uncovered: Certificate-Based Security Solutions for Next-Generation Networks.
Cisco Press, 2011.

Bibliografía IV


Carlos Munuera.
Steganography from a Coding Theory Point of View, pages 83–128.
WORLD SCIENTIFIC, 2013.


Satoshi Nakamoto.
Bitcoin: un sistema de dinero en efectivo electrónico peer-to-peer.
Technical report, bitcoin.org.
Traducido por @breathingdog.


National Institute of Standards and Technology (NIST).
DATA ENCRYPTION STANDARD (DES), October 1999.


National Institute of Standards and Technology (NIST).
ADVANCED ENCRYPTION STANDARD (AES), November 2001.


National Institute of Standards and Technology (NIST).
SECURE HASH STANDARD, August 2002.

Bibliografía V


National Institute of Standards and Technology (NIST).
Digital Signature Standard (DSS), July 2013.


Arvind Narayanan, Joseph Bonneau, Edward W. Felten, Andrew Miller, Steven Goldfeder, and Jeremy Clark.
Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction.
Princeton University Press, 2016.


Josef Pieprzyk, Thomas Hardjono, and Jennifer Seberry.
Fundamentals of Computer Security.
Springer, 2003.