

Cryptanalysis

Discrete Log Based Systems

John Manferdelli

JohnManferdelli@hotmail.com

© 2004-2020, John L. Manferdelli.

This material is provided without warranty of any kind including, without limitation, warranty of non-infringement or suitability for any purpose. This material is not guaranteed to be error free and is intended for instructional use only.

HMAC's concluded

- $\text{HMAC}(K, \text{text}) = H((K \oplus \text{opad}) || H((K \oplus \text{ipad}) || \text{text}))$
- H is a cryptographic hash like SHA-256
- ipad , the inner pad: the byte 0x36 repeated B times where B is key size
- opad , the outer pad: the byte 0x5c repeated B times
- Verification requires knowledge of K .

Discrete log based public key systems

Discrete Log

- If $b = a^x$, then $L_a(b) = x$. $L_a(y)$ is the discrete log function.
- If $g = b^x$, then $L_a(g) = xL_a(b)$. $L_a(b_1b_2) = L_a(b_1) + L_a(b_2)$
- **Discrete Log Problem (DLP):** Given p , prime, $a: \langle a \rangle = F_p^*$. $B \pmod{p}$, a , unknown, find $L_a(b)$.
- **Computational Diffie Hellman Problem (CDHP):** Given p , prime, $\langle a \rangle = F_p^*$. $a^a \pmod{p}$, $a^b \pmod{p}$, find $a^{ab} \pmod{p}$.
- **Theorem:** $\text{CDHP} \leq_p \text{DLP}$. If the factorization of $p-1$ is known and $f(p-1)$ is $O((\ln(p))^c)$ smooth then DLP and CDHP are equivalent.
- **Conclusion:** Exponentiation is a one-way trap-door function.

El Gamal cryptosystem

- Alice, the private keyholder, picks a large prime, p , where $p-1$ also has large prime divisors (say, $p = 2rq + 1$) and a generator, g , for F_p^* . $\langle g \rangle = F_p^*$. Alice also picks a random number, a (secret), and computes $A = g^a \pmod{p}$. Alice's public key is $\langle A, g, p \rangle$.
- To send a message, m , Bob picks a random b (his secret) and computes $B = g^b \pmod{p}$. Bob transmits $(B, mA^b) = (B, C)$.
- Alice decodes the message by computing $CB^{-a} = m$.
- Without knowing a , an adversary has to solve the Computational Diffie Hellman Problem to get m .
- Note: b must be random and never reused!

El Gamal Example

- Alice chooses
 - $p=919$. $g=7$.
 - $a=111$, $A=7^{111}=461 \pmod{919}$.
 - Alice's Public key is $\langle 919, 7, 461 \rangle$
- Bob wants to send $m=45$, picks $b=29$.
 - $B=7^{29}=788 \pmod{919}$, $461^{29}=902 \pmod{919}$,
 - $C=(45)(902)=154 \pmod{919}$.
 - Bob transmits $(788, 154)$.
- Alice computes $(788)^{-111}=902^{-1} \pmod{919}$.
 - $(54)(902)+(-53)(919)=1$. $54=902^{-1} \pmod{919}$
 - Calculates $m=(154)(54)=45 \pmod{919}$.

El Gamal Signature

- $\langle g \rangle = \mathbb{Z}_q^*$. A picks a random as in encryption.
- Signing: Signer picks k : $1 \leq k \leq p-2$ with $(k, p-1) = 1$ and publishes g^k . k is secret.
- $\text{Sig}_k(M, k) = (t, d)$
 - $t = g^k \pmod{p}$
 - $d = (M - gt)k^{-1} \pmod{p-1}$
- $\text{Ver}_k(M, t, d)$ iff $g^{kt} t^d = g^M \pmod{p}$
- Notes: It's important that M is a hash otherwise there is an existential forgery attack. It's important that k be different for every message otherwise adversary can solve for key.

Timing

- Finding g takes about $O(\lg(p)^3)$ operations, so does primality testing and raising g to the a power mod p .
- Encryption is also $O(\lg(p)^3)$ and so is decryption.
- Note that key generation is cheap but for safety, $p > w^2$, where w is the “computational power” of the adversary.

Finding generators (Gauss)

- Find a generator, g , for F_p^* , $n = p - 1 = p_1^{e_1} \cdot p_2^{e_2} \cdot \dots \cdot p_k^{e_k}$.

```
while(1) {  
    choose a random  $g \in G$   
    for( $i=1$ ;  $i \leq k$ ;  $k++$ ) {  
         $b = g^{n/p_i}$   
        if ( $b == 1$ )  
            break;  
    }  
    if( $i > k$ )  
        return  $g$   
}
```

- G has $\phi(n)$ generators. Using the lower bound for $\phi(n)$ the probability that g in line 2 is a generator is at least $1/(6 \ln \ln n)$

Attack on reused nonce

- Suppose Bob reuses b for two different messages m_1 and m_2 .
- An adversary, Eve, can see $\langle B, C_1 \rangle$ and $\langle B, C_2 \rangle$ where $C_i = Bm_i \pmod{p}$.
- Suppose Eve discovers m_1 .
- She can compute $m_2 = m_1 C_2 C_1^{-1} \pmod{p}$.
- Don't reuse b 's!

DSA

- Alice
 - $2^{159} < q < 2^{160}$, $2^{511+64t} < p < 2^{512+64t}$, $1 \leq t \leq 8$, $q | p-1$
 - Select primitive root $x \pmod{p}$; compute: $g = x^{(p-1)/q} \pmod{p}$
 - Picks a random, $1 < a < q-1$. $A = g^a \pmod{p}$
 - Public Key: (p, q, g, A) . Private Key: a .
- Signature Generation
 - Pick random k , $r = (g^k \pmod{p}) \pmod{q}$. Note : **k must be different for each signature.**
 - $s = k^{-1}(h(M) + ar) \pmod{q}$. Signature is (r, s)
- Verification
 - $u = s^{-1}h(x) \pmod{q}$, $v = (rs^{-1}) \pmod{q}$
 - Is $g^u A^v = r \pmod{p}$?
- Advantages over straight El Gamal
 - Verification is more efficient (2 exponentiations rather than 3)
 - Exponent is 160 bits not 768

Baby Step Giant Step --- Shanks

- $g^x = y(\text{mod } p)$.
- $m \approx \sqrt{p}$.
- Compute $g^{jm}, 0 \leq j < m$.
- Sort (j, g^{jm}) by second coordinate.
- Pick i at random, compute $yg^{-i}(\text{mod } p)$.
- If there is a match in the tables $yg^{-i} = g^{jm}(\text{mod } p)$ $yg^{-i} = g^{mj}$.
- $x = mj + i$ is the discrete log.

Baby Step Giant Step Example

- $p=193$. $\lfloor \sqrt{p} \rfloor=13$. $m=14$. $a=5$. $b=41$. Compute $\log_{193}(41)$.
- $2 \times 193 + (-77) \times 5 = 1$, $a^{-1} = 116$. $a^{-14} = 189 \pmod{193}$.

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a^j	5	25	125	46	37	185	153	186	158	18	90	64	127	56
ba^{-mj}	26	77	78	74	90	26	89	30	73	94	10	153	160	132

- So $ba^{-(14 \times 5)} = 90 = a^{11} \pmod{193}$.
- Thus $b = a^{14 \times 5 + 11} = a^{81} \pmod{193}$.
- $L_{193}(41) = 81$.

Discrete log Pollard r

- $x_{i+1} = f(x_i)$
 - $f(x_i) = bx_i$, if $x_i \in S_1$.
 - $f(x_i) = x_i^2$, if $x_i \in S_2$.
 - $f(x_i) = ax_i$, if $x_i \in S_3$.
- $x_i = a^{a[i]}b^{b[i]}$.
 - $a[i] = a[i]$, if $x_i \in S_1$.
 - $a[i] = 2a[i]$, if $x_i \in S_2$.
 - $a[i] = a[i] + 1$, if $x_i \in S_3$.
 - $b[i] = b[i] + 1$, if $x_i \in S_1$.
 - $b[i] = 2b[i]$, if $x_i \in S_2$.
 - $b[i] = b[i]$, if $x_i \in S_3$.
- $x_{2i} = x_i \rightarrow a_{2i} - a_i = L_a(b)(b_{2i} - b_i)$

Pollard ρ example

- $p=229$, $n=191$, $b=228$, $a=2$. $L_2(228)=110$

i	x_i	a_i	b_i
1	228	0	1
2	279	0	2
3	92	0	4
4	184	1	4
5	205	1	5
6	14	1	6
7	28	2	6
8	256	2	7
9	152	2	8
10	304	3	8
11	372	3	9
12	121	6	18
13	12	6	19
14	144	12	38

i	x_{2i}	a_{2i}	b_{2i}
1	279	0	2
2	184	1	4
3	14	1	6
4	256	2	7
5	304	3	8
6	121	6	38
7	144	12	152
8	235	48	154
9	72	48	118
10	14	96	119
11	256	97	120
12	304	98	51
13	121	5	104
14	144	10	163

- $x_{14} = x_{28}$, $(b_{14} - b_{28}) = 125 \pmod{191}$, $L_2(228) = 125^{-1} (a_{28} - a_{14}) = 110$.

Pohlig-Hellman

- $p - 1 = \prod_i q_i^{r_i}$.
- Solve $a^x = y \pmod p$ for $x \pmod{q_i^{r[i]}}$ and use Chinese Remainder Theorem.
- $x = x_0 + x_1 q + x_2 q^2 + \dots + x_{r[i]-1} q^{r[i]-1}$.
- $x (p-1)/q = x_0 (p-1)/q + (p-1)(\dots)$
- So $b^{(p-1)/q} = a^{x[0](p-1)/q}$. Solve for x_0 .
- Then put $g = ba^{-x[0]}$ and solve $g^{(p-1)/(q \times q)} = a^{x[1](p-1)/q}$.
- This costs $O(\sum_{i=1}^r e_i (\lg(n) + \sqrt{q_i}))$.

Pohlig-Hellman example

- $p=251$. $a=71$, $b=210$, $\langle a \rangle = F_{251}^*$. $n=250=2 \times 5^3$.
- $L_{71}(210) = 1 \pmod{2}$.
- $x = x_0 + x_1 5 + x_2 5^2$.
- So $a^{n/5} = 71^{20}$. $b^{n/5} = 210^{20} = 149$.
 - $x_0 = L_{20}(149) = 2$.
 - $x_1 = 4$
 - $x_2 = 2$
- $x = 2 + 4 \times 5 + 2 \times 25 = 72 \pmod{125}$
- Applying CRT: $L_{71}(210) = 197$.

Index Calculus

- $g^x = y \pmod{p}$. $B = (p_1, p_2, \dots, p_k)$.
- Precompute
 - $g^{x_j} = p_1^{a_1} \cdot p_2^{a_2} \cdot \dots \cdot p_k^{a_k}$
 - $x_j = a_{1j} \log_g(p_1) + a_{2j} \log_g(p_2) + \dots + a_{kj} \log_g(p_k)$
 - If you get enough of these, you can solve for the $\log_g(p_i)$
- Solve
 - Pick s at random and compute: $yg^s = p_1^{c_1} \cdot p_2^{c_2} \cdot \dots \cdot p_k^{c_k}$
 - $\log_g(y) + s = c_1 \log_g(p_1) + c_2 \log_g(p_2) + \dots + c_k \log_g(p_k)$
- This takes $O(e^{(1+\ln(p))\ln(\ln(p))})$ time.
- LaMacchia and Odlyzko used Gaussian integer index calculus variant to attack discrete log.

Index Calculus Example

- $p=229$. $a=6$. $\langle a \rangle = F_{229}^*$. $n=228$. $b=13$. $S=\{2,3,5,7,11\}$.
- Step 1
 1. $6^{100} \pmod{229} = 180 = 2^2 \times 3^2 \times 5^1 \times 7^0 \times 11^0$.
 2. $6^{18} \pmod{229} = 176 = 2^4 \times 3^0 \times 5^0 \times 7^0 \times 11^1$.
 3. $6^{12} \pmod{229} = 165 = 2^0 \times 3^1 \times 5^1 \times 7^0 \times 11^1$.
 4. $6^{62} \pmod{229} = 154 = 2^1 \times 3^0 \times 5^0 \times 7^1 \times 11^1$.
 5. $6^{143} \pmod{229} = 198 = 2^1 \times 3^2 \times 5^0 \times 7^0 \times 11^1$.
 6. $6^{206} \pmod{229} = 210 = 2^1 \times 3^1 \times 5^1 \times 7^1 \times 11^0$.
- Taking $L_a()$ of both sides, we get:
 1. $100 = 2 L_a(2) + 2 L_a(3) + L_a(5) \pmod{228}$
 2. $18 = 4 L_a(2) + L_a(11) \pmod{228}$
 3. $12 = L_a(3) + L_a(5) + L_a(11) \pmod{228}$
 4. $62 = L_a(2) + L_a(7) + L_a(11) \pmod{228}$
 5. $143 = L_a(2) + 2 L_a(3) + L_a(11) \pmod{228}$
 6. $206 = L_a(2) + L_a(3) + L_a(5) + L_a(7) \pmod{228}$

Index Calculus example - continued

- Review

- $p=229$. $a=6$. $\langle a \rangle = F_{229}^*$. $n=228$. Solving, we got:
- $L_a(2) = 21 \pmod{228}$
- $L_a(3) = 208 \pmod{228}$
- $L_a(5) = 98 \pmod{228}$
- $L_a(7) = 107 \pmod{228}$
- $L_a(11) = 162 \pmod{228}$

Step 2:

- Recall $b=13$. Pick $k=77$
- $13 \times 6^{77} = 147 = 3 \times 7^2 \pmod{229}$
- $L_6(13) = (L_6(3) + 2L_6(7) - 77) = 117 \pmod{228}$

Diffie Hellman key exchange

Alice

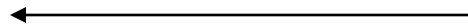
Bob

A1: $s = \min(p \text{ size}),$
 $N_a \text{ in } \{0, \dots 2^{256}-1\}$

s, N_a



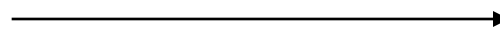
$(p, q, g), X = g^x,$
 Auth_B



A2: Check $(p, q, g) X,$
 $\text{Auth}_B,$ pick y in
 $\{0, \dots q-1\}$

$K = X^y$

$Y = g^y, \text{Auth}_A$



B1: Choose $(p, q, g),$
 $x \text{ in } \{0, \dots 2^{256}-1\}$

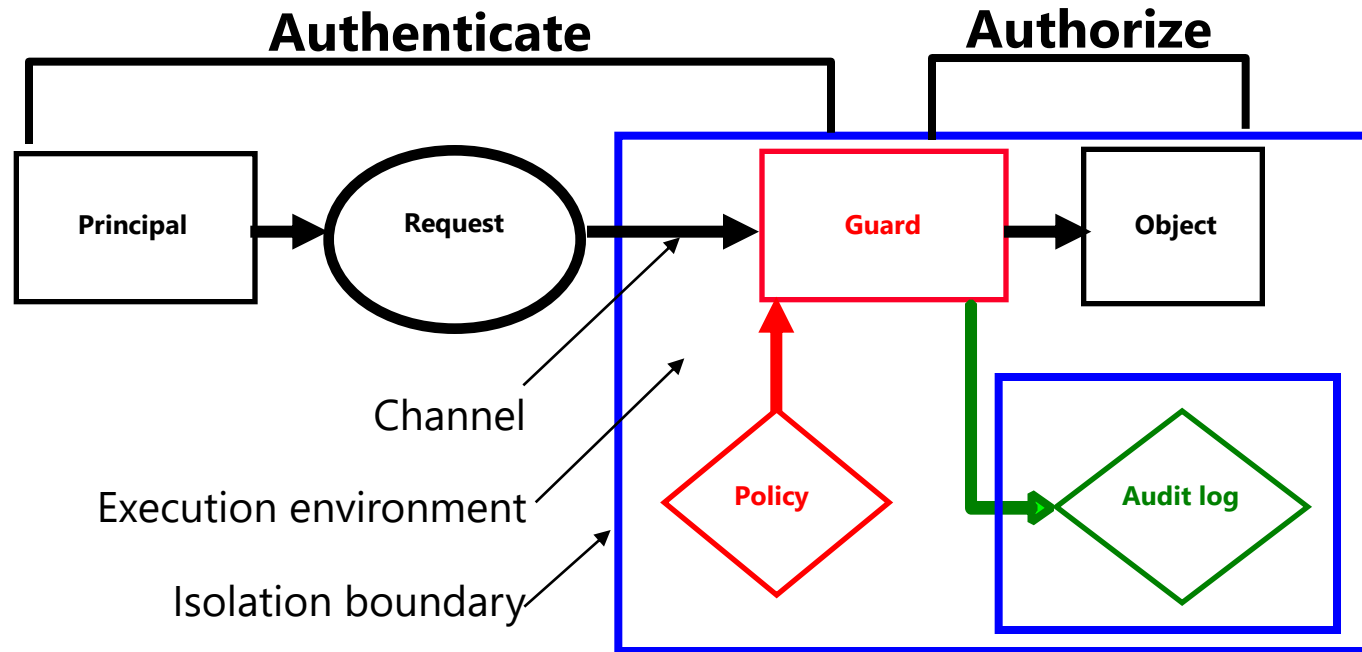
B2: Check Y, Auth_A

$K = Y^x$

DH key exchange example

- $p=3547, g=2$.
- Alice: $a=7$.
- Bob: $b=17$.
- $A \rightarrow B_1$: $A=128 (=2^7)$, $\text{Sign}_A(\text{SHA-2}(128 || r_1))$
- $B \rightarrow A_1$: $B=3380 (=2^{17})$, $\text{Sign}_B(\text{SHA-2}(3380 || r_2))$
- $K = 128^{17} = 3380^7 = 362$.

Access Control: authentication and authorization



- Authentication is process of identifying a security principal. Here are some ways:
 - Login/password or smart card/pin (user)
 - Cryptographic Hash (program)
 - Ability to decrypt (channel)

Authentication

- When logging on to a computer you enter
 - user name and
 - password
- The first step is called **identification**. You announce who you are.
- The second step is called **authentication**. You prove that you are who you claim to be.
- To distinguish this type of ‘authentication’ from other interpretations, we may refer specifically to **entity authentication**: The process of verifying a claimed identity.

Authentication

Login: jlm

Password: ****

Welcome John Manferdelli

>

Problems with Passwords

- Authentication by password is widely accepted and not too difficult to implement.
- Managing password security can be quite expensive; obtaining a valid password is a common way of gaining unauthorised access to a computer system.
- Typical issues
 - how to get the password to the user,
 - forgotten passwords,
 - password guessing,
 - password spoofing,
 - compromise of the password file.

Guessing Passwords

- Exhaustive search (brute force): Try all possible combinations of valid symbols up to a certain length.
- Intelligent search: search through a restricted name space, e.g. passwords that are somehow associated with a user like name, names of friends and relatives, car brand, car registration number, phone number,..., or try passwords that are generally popular.
- Typical example for the second approach: dictionary attack trying all passwords from an on-line dictionary.
- You cannot prevent an attacker from accidentally guessing a valid password, but you can try to reduce the probability of a password compromise.

Password Salting

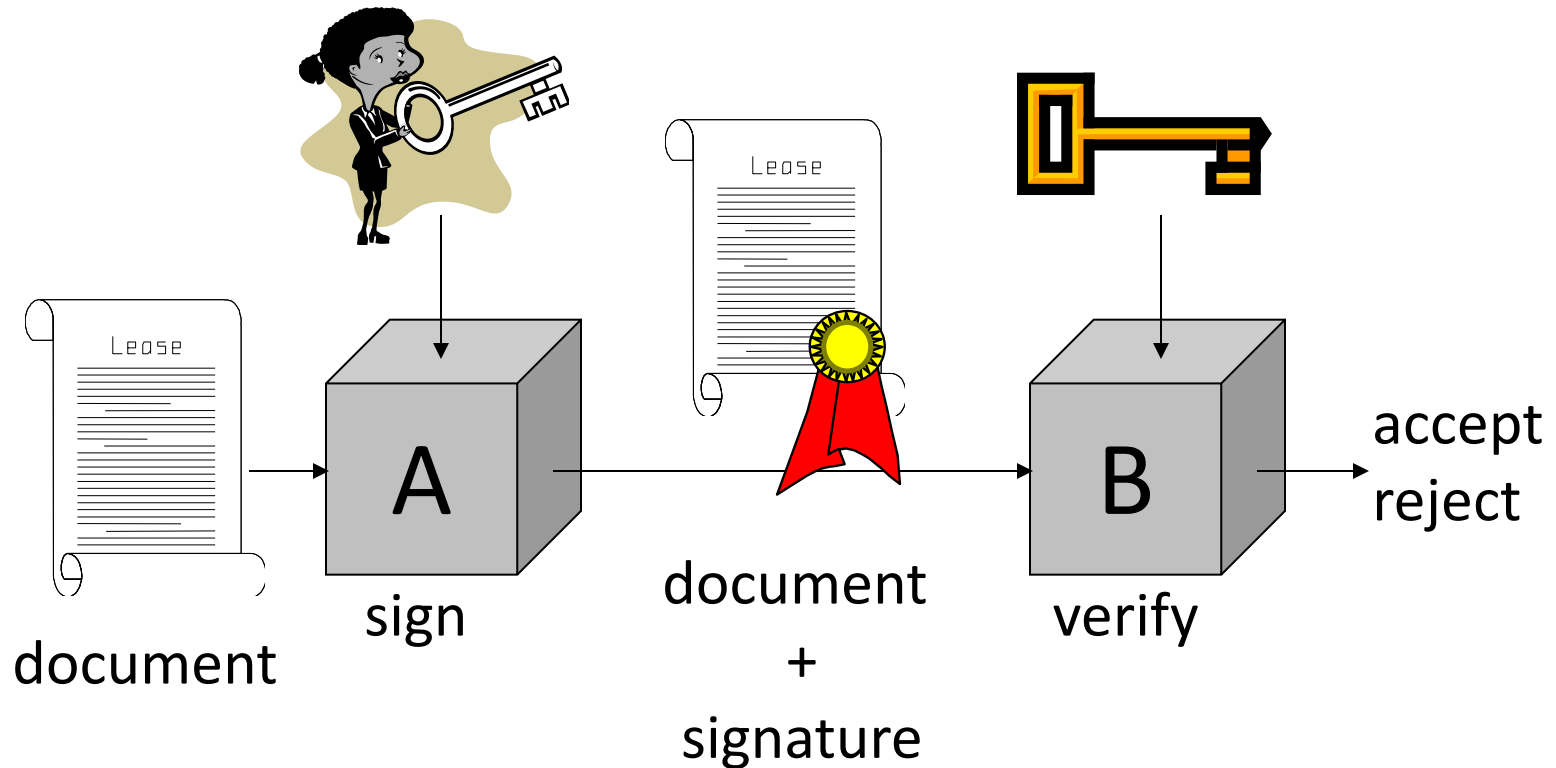
- To slow down dictionary attacks, a **salt** can be appended to the password before encryption and stored with the encrypted password.
 - If two users have the same password, they will now have different entries in the file of encrypted passwords.
 - Example: Unix uses a 12 bit salt.

Access Control Matrix

- Capabilities:
 - access rights are stored with the subject
 - rows of the access control matrix
- Access Control Lists (ACLs)
 - access rights are stored with the object.
 - columns of the access control matrix.

	bill.doc	edit.exe	fun.com
Alice	-	{exec}	{exec,read}
Bob	{read,write}	{exec}	{exec,read,write}

Digital signatures



Slide from Dieter Gollmann

Digital Signatures

- A has a public verification key and a private signature key (\rightarrow public key cryptography).
- A uses her private key to compute her signature on document m .
- B uses a public verification key to check the signature on a document m he receives.
- This provides non-repudiation.
- Signature algorithm= hash+padding+private key operation

Bleichenbacher Attack on PKCS1

- Chosen-ciphertext attack.
- RSA PKCS #1 v1.5 : $c = (00 || 02 || r || 0 || m)^e \bmod n$
- Attacker can test if 16 MSBs of plaintext = '02'.
- Attack: to decrypt a given ciphertext C do:
 - Pick $r \in \mathbb{Z}_n$.
 - Compute $C' = r^e \cdot C = (r \cdot \text{PKCS1}(M))^e$.
 - Send C' to oracle and use response.

Side-Channel Attacks

- Some attack vectors ...
 - Fault Attacks
 - Timing Attacks
 - Cache Attacks
 - Power Analysis
 - Electromagnetic Emissions
 - Acoustic Emissions

End

Berlekamp factorization

- $f(x) = \prod_{i=1}^t f_i(x)$ over F_p , $\deg(f(x)) = n$. $f_i(x)$ irreducible.
 $F = \{f(x)\};$
for($i=1; i < n; i++$)

$$x^{iq} = \sum_{j=0}^{n-1} q_{ij} x^j \pmod{f(x)}, q_{ij} \in F_p.$$
Find basis $\langle v_1, \dots, v_t \rangle$ of null space of $(Q - I_n);$
// $w = w_0, \dots, w_{n-1}$. $w(x) = w_0 + w_1 x + \dots + w_{n-1} x^{n-1}$
for($i=1; i \leq t; i++$) {
for ($h(x) \in F, \deg(h) > 1;$) {
Compute $(h(x), v_i(x) - a), a \in F_p;$
Replace $h(x)$ in F with these;
}
return (F);
- $O(n^3 + t p n^2)$, $t = \#$ irreducible factors. Can be reduced to $O(n^3 + t \lg(p) n^2)$.

Berlekamp factorization example

- Factor x^7-1 over $GF(2)$.

1	0	0	0	0	0	0	1	1
0	0	1	0	0	0	0	x	x^2
0	0	0	0	1	0	0	x^2	x^4
0	0	0	0	0	0	1	x^3	x^6
0	1	0	0	0	0	0	x^4	x^1
0	0	0	1	0	0	0	x^5	x^3
0	0	0	0	0	1	0	x^6	x^5

=

- Adding and solving get:
 - 1
 - $x^4+x^2+x = x(x^3+x+1)$
 - $x^6+x^5+x^3 = x^3(x^3+x^2+1)$
 - Dividing into x^7-1 , we get: $(x+1)$