Cryptocomplexity II

Kryptokomplexität II

Sommersemester 2024

Chapter 3: ElGamal's Protocols

Dozent: Prof. Dr. J. Rothe

The Protocols of ElGamal

- In 1985, Taher ElGamal modified the Diffie–Hellman protocol to
  - a public-key cryptosystem and
  - a digital signature scheme.

- A particularly efficient variant of this protocol, due to an ingenious idea of Schnorr (1990), has been adopted in 1993 as the United States *Digital Signature Standard (DSS)* (specified in FIPS-186):
  - The Digital Signature Algorithm (DSA) is covered by U.S. Patent 5,231,668 and attributed to David W. Kravitz, a former NSA employee.
  - Claus P. Schnorr claims that his U.S. Patent 4,995,082 (expired) covered DSA; this claim is disputed.

- The security of both protocols rests on the difficulty of computing discrete logarithms.

ElGamal's Public-Key Cryptosystem

Step	Alice	Erich	Bob
1	Alice and Bob agree upon a large prime p and a primitive element γ of p; p and γ are public		
2			chooses a large random number b as his private key and computes $\beta = \gamma^b \bmod p$
3		$\Leftarrow \beta$	
4	chooses a large random number a and encrypts the message m by: $\alpha_1 = \gamma^a \bmod p$ and $\alpha_2 = m\beta^a \bmod p$		
5		$(\alpha_1, \alpha_2) \Rightarrow$	
6			decrypts by computing $\alpha_2 (\alpha_1)^{-b} \bmod p$

ElGamal's Public-Key Cryptosystem

1. **Preparation.** As in the Diffie–Hellman protocol, Alice and Bob agree upon
   - a large prime number p such that the discrete logarithm problem is intractable in $\mathbb{Z}_p^*$ and
   - a primitive element γ of p.
   - Both p and γ are public.

2. **Key Generation.** Bob generates his private key b at random and computes his public key by

$$\beta = \gamma^b \bmod p.$$

3. **Communication.** Bob's public key β is now known to Alice.

ElGamal's Public-Key Cryptosystem

4. **Encryption.**
   - As usual, messages are encoded block-wise, where any block is represented by an element of the plaintext space $\mathbb{Z}_p^*$.
   - Suppose that Alice wants to send the message block $m \in \mathbb{Z}_p^*$ to Bob.
   - The ciphertext space is $\mathbb{Z}_p^* \times \mathbb{Z}_p^*$, and the two components of the ciphertext $c = (\alpha_1, \alpha_2)$ encrypting m are computed by:

$$\alpha_1 = \gamma^a \bmod p \quad (1)$$

$$\alpha_2 = m\beta^a \bmod p. \quad (2)$$

   - The encryption function $E_{(p,\gamma,\beta,a)}(m) = (\alpha_1, \alpha_2)$ is defined according to (1) and (2).

ElGamal's Public-Key Cryptosystem

- Alice "masks" her plaintext m by multiplying it by her "Diffie–Hellman key"

$$\beta^a \equiv \gamma^{ba} \bmod p.$$

- The value of $\alpha_1 = \gamma^a \bmod p$ is also part of the ciphertext in order to allow decryption by the legitimate receiver Bob.

1. **Communication.** Alice sends the ciphertext $c = (\alpha_1, \alpha_2)$ to Bob.

ElGamal's Public-Key Cryptosystem

5. **Decryption.**
   - The decryption function is given by

$$D_{(p,\gamma,b)}(\alpha_1, \alpha_2) = \alpha_2 (\alpha_1)^{-b} \bmod p. \tag{3}$$

   - According to (3), Bob uses his private key b to first compute

$$\gamma^{-ab} \bmod p$$

     from $\alpha_1 = \gamma^a \bmod p$.

   - Then, multiplying α_2 by γ^{-ab}, he removes the "mask" β^a from the plaintext.

   - Summing up, Bob decrypts the ciphertext c by computing

$$\alpha_2 (\alpha_1)^{-b} \equiv m\beta^a (\gamma^a)^{-b} \equiv m\gamma^{ba}\gamma^{-ab} \equiv m \bmod p$$

     and thus obtains the original plaintext m.

ElGamal's Public-Key Cryptosystem

- ElGamal's system modifies the Diffie–Hellman protocol in the following way:
  - While in the Diffie–Hellman scheme Alice and Bob *simultaneously* compute and send their "partial keys" α and β, respectively,
  - they do so *sequentially* in the ElGamal protocol.
  - That is, Alice must wait for Bob's value β to be able to compute her second component of the ciphertext, α_2, in which her message m is "masked" by β^a.

- Another difference between the two protocols:
  - Bob generates his public key β once and for all in the ElGamal protocol. Thus, he can use β for more than one communication.
  - However, Alice has to generate her secret exponent a and thus her $\alpha_1 = \gamma^a \bmod p$ anew again and again every time she communicates with Bob, just as in the Diffie–Hellman protocol.

ElGamal's Public-Key Cryptosystem: Example

Example (ElGamal's Public-Key Cryptosystem)

Alice and Bob choose

- the prime number $p = 101$ and
- the primitive element $\gamma = 8$ of 101.

(Check: $\gamma^{(p-1)/q} \not\equiv 1 \bmod p$ for all prime divisors q of $p-1 = 100$:

- $8^{100/2} = 8^{50} \equiv 100 \bmod 101$;
- $8^{100/5} = 8^{20} \equiv 87 \bmod 101$.)

Bob chooses $b = 12$ as his private key and computes the public key

$$\beta = \gamma^b = 8^{12} \bmod 101 = 78.$$

ElGamal's Public-Key Cryptosystem: Example

Example (ElGamal's Public-Key Cryptosystem: continued)

Alice chooses her private exponent $a = 33$ and computes key

$$\beta^a = 78^{33} \bmod 101 = 92.$$

To encrypt the plaintext $m = 53$, she computes

$$\alpha_1 = \gamma^a \bmod p = 8^{33} \bmod 101 = 51$$
$$\alpha_2 = m\beta^a \bmod p = 53 \cdot 92 \bmod 101 = 28$$

and sends $c = (\alpha_1, \alpha_2) = (51, 28)$ to Bob.

ElGamal's Public-Key Cryptosystem: Example

Example (ElGamal's Public-Key Cryptosystem: continued)

Bob decrypts c

$$\begin{aligned} \alpha_2(\alpha_1)^{-b} &= 28(51)^{-12} \\ &\equiv 28(51^{-1})^{12} \\ &\equiv 28 \cdot 2^{12} \\ &\equiv 28 \cdot 56 \\ &\equiv 53 \bmod 101 \end{aligned}$$

and obtains the original plaintext $m = 53$.

ElGamal's Digital Signature Scheme

Step	Alice	Erich	Bob
1	Alice and Bob agree upon a large prime p and a primitive element γ of p; p and γ are public		
2			chooses two large random numbers b, s with $\gcd(s, p-1) = 1$, and computes his signature for message m by $\text{sig}_B(m) = (\sigma, \rho)$, $\beta = \gamma^b \bmod p$, $\sigma = \gamma^s \bmod p$, $\rho = (m - b\sigma)s^{-1} \bmod (p-1)$
3		$\Leftarrow \langle m, \beta, \text{sig}_B(m) \rangle$	
4	verifies Bob's signature by checking $\gamma^m \equiv \beta^\sigma \sigma^\rho \bmod p$		

ElGamal's Digital Signature Scheme

- **Preparation.** Alice and Bob agree on
  - a large prime number p, chosen so that the discrete logarithm problem is infeasible in $\mathbb{Z}_p^*$, and
  - on a primitive element γ of p.
  - Both p and γ are public.

- **Signing the message.**
  - Suppose that Bob wants to send Alice some message m.
  - As in the ElGamal cryptosystem, Bob chooses his private exponent b and computes

$$\beta = \gamma^b \bmod p.$$

  - In addition, he now chooses a secret number s coprime with $p-1$, keeping b and s secret.

ElGamal's Digital Signature Scheme

- **Signing the message (continued).**
  - To sign m, Bob first computes

$$\sigma = \gamma^s \bmod p$$

    and a solution ρ to the congruence

$$b\sigma + s\rho \equiv m \bmod p-1 \tag{4}$$

    using the extended algorithm of Euclid.
  - Then, his signature for m is defined by

$$\mathrm{sig}_B(m) = (\sigma, \rho).$$

- **Communication.** Along with his message m, Bob sends his digital signature

$$\mathrm{sig}_B(m) = (\sigma, \rho)$$

  and the value β to Alice.

ElGamal's Digital Signature Scheme

- **Verifying the signature.** Alice checks the validity of the signature by verifying the congruence

$$\gamma^m \equiv \beta^\sigma \sigma^\rho \bmod p. \tag{5}$$

- By Fermat's Little Theorem and by (4), we have that

$$\gamma^m \equiv \gamma^{b\sigma+s\rho} \equiv \beta^\sigma \sigma^\rho \bmod p.$$

Thus, as desired, (5) verifies correctly that Bob's signature is valid, which shows that the ElGamal digital signature protocol works.

ElGamal's Digital Signature Scheme

Example

- Let $p = 1367$ be a given prime number, and let $\gamma = 5$ be a given primitive element of 1367. Suppose that Bob chooses the private exponents $b = 513$ and $s = 129$; note that $\gcd(129, 1366) = 1$.
- First, Bob computes

$$\beta = 5^{513} \bmod 1367 = 855 \quad \text{and} \quad \sigma = 5^{129} \bmod 1367 = 1180.$$

- Suppose that Bob wants to sign the message $m = 457$. Bob has to solve the congruence

$$513 \cdot 1180 + 129\rho \equiv 457 \bmod 1366$$

for ρ.

ElGamal's Digital Signature Scheme

Example (continued)

- Using the extended algorithm of Euclid, he determines the inverse element $s^{-1} = 593$ of $s = 129$ modulo 1366, and thus he obtains the solution

$$\rho = (457 - 513 \cdot 1180)593 \bmod 1366 = 955.$$

- Now, Bob's signature for $m = 457$ is given by

$$\mathrm{sig}_B(457) = (1180, 955)$$

and he transfers the triple $\langle 457, 855, (1180, 955)\rangle$ to Alice.

ElGamal's Digital Signature Scheme

Example (continued)

- On the other side of town, Alice checks whether the signature is valid by verifying the congruence

$$5^{457} \equiv 1280 \equiv 749 \cdot 750 \equiv 855^{1180} \cdot 1180^{955} \bmod 1367. \tag{6}$$

- As usual, Alice employs the "square-and-multiply" algorithm to compute the values γ^m, β^σ, and σ^ρ in the arithmetics modulo p:

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}
5	25	625	1030	108	728	955	236	1016		
855	1047	1242	588	1260	513	705	804	1192	551	127
1180	794	249	486	1072	904	1117	985	1022	96	

ElGamal's Digital Signature Scheme

Example (continued)

- The gray boxes of this table contain the values to be multiplied according to the binary expansion of the exponents:

$$\begin{aligned} m &= 457 = 2^0 + 2^3 + 2^6 + 2^7 + 2^8; \\ \sigma &= 1180 = 2^2 + 2^3 + 2^4 + 2^7 + 2^{10}; \\ \rho &= 955 = 2^0 + 2^1 + 2^3 + 2^4 + 2^5 + 2^7 + 2^8 + 2^9. \end{aligned}$$

- This yields:

$$\begin{aligned} \gamma^m \bmod p &= 1280 \\ \beta^\sigma \bmod p &= 749 \\ \sigma^\rho \bmod p &= 750 \end{aligned}$$

Security of ElGamal's Protocols

- Just as with the Diffie–Hellman protocol, the security of the ElGamal cryptosystem and of the ElGamal digital signature scheme relies on the hardness of the discrete logarithm problem:

  If Erich can compute discrete logarithms efficiently, then he can break, for example, the ElGamal cryptosystem by computing Bob's private key

  $$b = \log_\gamma \beta \bmod (p-1)$$

  from Bob's public key β and the public prime p with its public primitive element γ.

Security of ElGamal's Protocols

- On the other hand, it is not known if
  - computing discrete logarithms and
  - breaking either of the ElGamal protocols

  are equally hard problems.

- However, it can be shown that
  - breaking the ElGamal public-key cryptosystem

    is computationally equivalent to
  - the Diffie–Hellman problem (DIFFIE-HELLMAN).

Problem of Breaking ElGamal

Definition

Define the (functional) *problem of breaking ElGamal*, denoted by BREAK-ELGAMAL, as follows: Given $\langle p, \gamma, \beta, \alpha_1, \alpha_2 \rangle$, where

- p is a prime number,
- γ is a primitive element of p, and
- β, α_1, and α_2 are defined as in the ElGamal system for any message m,

compute m.

Security of ElGamal's Public-Key Cryptosystem

Theorem

The problem of breaking ElGamal and the Diffie–Hellman problem are equivalent under polynomial-time Turing reductions. That is,

1. $\text{BREAK-ELGAMAL} \in \text{FP}^{\text{DIFFIE-HELLMAN}}$ *and*
2. $\text{DIFFIE-HELLMAN} \in \text{FP}^{\text{BREAK-ELGAMAL}}$.

Proof:

1. Suppose that eavesdropper Erich has an algorithm, D, for solving the Diffie–Hellman problem.

   He wants to use D to break ElGamal's cryptosystem.

   Let p be a prime number, and let γ be a primitive element of p.

Security of ElGamal's Public-Key Cryptosystem

As in ElGamal's cryptosystem, for any message m, let β, α_1, and α_2 be the transmitted values, which Erich knows.

On input $\langle p, \gamma, \beta, \alpha_1, \alpha_2 \rangle$, he wishes to compute the corresponding message m.

Looking at ElGamal's cryptosystem, note that $\alpha_1 = \gamma^a \bmod p$ and $\beta = \gamma^b \bmod p$.

Using his algorithm D, Erich can compute $\gamma^{ab} \bmod p$ from α_1 and β.

Note further that

$$\alpha_2 = m\beta^a \equiv m\gamma^{ab} \bmod p.$$

Hence, using the extended algorithm of Euclid, Erich can recover the message m by computing $\alpha_2 \gamma^{-ab} \bmod p = m$.

Security of ElGamal's Public-Key Cryptosystem

2. Conversely, suppose that Erich has an algorithm, E, for breaking the ElGamal cryptosystem.

   Let p, γ, β, α_1, and α_2 be given as in ElGamal for an arbitrary message m. Using E, Erich can determine m from $\langle p, \gamma, \beta, \alpha_1, \alpha_2 \rangle$.

   To solve the Diffie–Hellman problem, given $\alpha_1 = \gamma^a \bmod p$ and $\beta = \gamma^b \bmod p$, he runs E on input $\langle p, \gamma, \beta, \alpha_1, 1 \rangle$ for the specific value of $\alpha_2 = 1$, obtaining some corresponding message m.

   It follows that

   $$m\beta^a \equiv m\gamma^{ab} \equiv 1 \bmod p.$$

   Thus, in order to determine $\gamma^{ab} = m^{-1} \bmod p$, it is enough to compute the inverse element of m modulo p, using the extended algorithm of Euclid. ❑

Bit Security of Discrete Logarithms

Definition

Define the (functional) *discrete logarithm bit problem*, denoted by DLOGBIT, as follows: Given $\langle p, \gamma, \alpha, i \rangle$, where

- p is a prime number,
- γ is a primitive element of p,
- $\alpha \in \mathbb{Z}_p^*$, and
- i is an integer with $1 \leq i \leq \lceil \log(p-1) \rceil$,

compute the i^{th} least significant bit in the binary representation of

$$\log_\gamma \alpha \bmod (p-1).$$

Bit Security of Discrete Logarithms

Example

- In a previous example, Shanks' algorithm was used to compute

$$\log_2 47 \bmod 100 = 58.$$

- Every element of $\mathbb{Z}_{101}^*$ can be represented in binary using no more than $\lceil \log 100 \rceil = 7$ bits.

- In particular, since

$$58 = 2^5 + 2^4 + 2^3 + 2^1,$$

the binary representation of 58 is $\texttt{bin}(58) = 111010$ and has six bits, dropping leading zeros.

The least significant bit of $\texttt{bin}(58)$ is the rightmost zero.

Bit Security of Discrete Logarithms

Example (continued)

- In general, the least significant bit of $\texttt{bin}(n)$ is the coefficient of 2^0 in the binary expansion of n.
- This bit determines the parity of n:
  - it is one if n is odd, and
  - it is zero if n is even.
- Suppose that an instance

$$\langle p, \gamma, \alpha, i \rangle = \langle 101, 2, 47, i \rangle$$

of DLOGBIT is given for $1 \leq i \leq 7$.

Bit Security of Discrete Logarithms

Example (continued)

- The following table shows the function values of $\text{DLOGBIT}(\langle 101, 2, 47, i \rangle)$ for the possible values of i, where leading zeros are not being dropped.

i	7	6	5	4	3	2	1
$\text{DLOGBIT}(\langle 101, 2, 47, i \rangle)$	0	1	1	1	0	1	0

Table: An instance of the discrete logarithm bit problem

Reminder: Quadratic Residue and Nonresidue

Definition

- For $n \in \mathbb{N}$, an element $x \in \mathbb{Z}_n^*$ is said to be a *quadratic residue modulo n* if there exists some $w \in \mathbb{Z}_n$ such that

$$x \equiv w^2 \bmod n.$$

- Otherwise, x is said to be a *quadratic nonresidue modulo n*.
- If $x = 1$, such a w is said to be a *square root of 1 modulo n*.
- Define the decision problems

$$\begin{aligned} \mathrm{QR} &= \{(x,n) \,|\, x \in \mathbb{Z}_n^*,\ n \in \mathbb{N}, \text{ and } x \text{ is a quadratic residue mod } n\}; \\ \mathrm{QNR} &= \{(x,n) \,|\, x \in \mathbb{Z}_n^*,\ n \in \mathbb{N}, \text{ and } x \text{ is a quadratic nonresidue mod } n\}, \end{aligned}$$

where x and n are represented in binary.

Reminder: Quadratic Residue and Nonresidue

Alternatively, we define the set of quadratic residues for a given $n \in \mathbb{N}$ by

$$\mathrm{QR}_n = \{w^2 \bmod n \,|\, w \in \mathbb{Z}_n^*\}.$$

Example (quadratic residue)

- Let $n = 13$.

w	1	2	3	4	5	6	7	8	9	10	11	12
$w^2 \bmod 13$	1	4	9	3	12	10	10	12	3	9	4	1

  $\mathrm{QR}_{13} = \{1, 3, 4, 9, 10, 12\}$.

- $\mathrm{QR}_{26} = \{1, 3, 9, 17, 23, 25\}$.
- $\mathrm{QR}_{27} = \{1, 4, 7, 10, 13, 16, 19, 22, 25\}$.

Reminder: Facts About Square Roots of 1 modulo n

- Trivially, 1 and $n-1$ are always square roots of 1 modulo n:

$$1^2 \equiv 1 \bmod n \quad \text{and} \quad (n-1)^2 \equiv (-1)^2 \equiv 1 \bmod n.$$

- If n is a prime number, then it has no square roots of 1 modulo n other than the trivial ones: $a^2 \equiv 1 \bmod n$ implies

$$(a+1)(a-1) = a^2 - 1 \equiv 0 \bmod n.$$

Thus n divides $(a+1)(a-1)$.

Since n is prime, n divides $a+1$ or $a-1$.

Hence, $a \equiv n-1 \bmod n$ or $a \equiv n+1 \equiv 1 \bmod n$.

- Hence, if n has a nontrivial square root of 1 modulo n, then n must be composite.

Reminder: Facts About Square Roots of 1 modulo n

- Conversely, if $n = p_1 p_2 \cdots p_k$ is composite, where the p_i are odd prime numbers, then the Chinese Remainder Theorem can be applied to show that n has exactly 2^k square roots of 1 modulo n, namely all numbers a, $1 \leq a \leq n$, satisfying

$$a \bmod p_i \in \{1, p_i - 1\}, \quad 1 \leq i \leq k.$$

- Thus, trying to find nontrivial square roots of 1 modulo n by randomly picking a number a is hopeless, unless n happens to have extraordinarily many prime factors.

Reminder: Facts About Square Roots of 1 modulo n

Example (nontrivial square roots of 1 modulo n)

1. Consider the composite number $n = 143 = 11 \cdot 13$.

   Since 143 has two prime factors, there are four square roots of 1 modulo 143, namely 1, 12, 131, and 142.

   The nontrivial square roots of 1 modulo 143 are 12 and 131.

   In this example, the square roots of 1 modulo 143 happen to be just the Fermat liars for 143. In general, however, this is not the case.

2. $n = 91 = 7 \cdot 13$ has the square roots: 1, 27, 64, and 90:

$$1 \equiv 64 \bmod 7 \quad \text{and} \quad 27 \equiv 90 \equiv 6 \bmod 7$$

$$1 \equiv 27 \bmod 13 \quad \text{and} \quad 64 \equiv 90 \equiv 12 \bmod 13.$$

Computing the Parity of Discrete Logarithms by Euler

Theorem (Euler's criterion)

Let p be an odd prime number. Then, x is a quadratic residue modulo p if and only if

$$x^{(p-1)/2} \equiv 1 \bmod p.$$

Proof: Suppose that x is a quadratic residue modulo p, i.e.,

$$x \equiv w^2 \bmod p$$

for some $w \in \mathbb{Z}_p^*$.

By Fermat's Little Theorem, $w^{p-1} \equiv 1 \bmod p$. Thus,

$$x^{(p-1)/2} \equiv \left(w^2\right)^{(p-1)/2} \equiv w^{p-1} \equiv 1 \bmod p.$$

Computing the Parity of Discrete Logarithms by Euler

Conversely, suppose that $x^{(p-1)/2} \equiv 1 \bmod p$.

Let γ be a primitive element modulo p.

Then we have $x \equiv \gamma^i \bmod p$ for some i.

It follows that

$$x^{(p-1)/2} \equiv \left(\gamma^i\right)^{(p-1)/2} \equiv \gamma^{i(p-1)/2} \equiv 1 \bmod p.$$

Since γ has the order $p-1$, it follows that $p-1$ divides $i(p-1)/2$.

Hence, i is even, and the two square roots of x are $\pm\gamma^{i/2}$. ❑

Using this result, we now show that the discrete logarithm bit problem can be efficiently solved for instances with $i = 1$.

Computing the Parity of Discrete Logarithms by Euler

Theorem

If $\langle p, \gamma, \alpha, 1 \rangle$ is an instance of the discrete logarithm bit problem, then $\text{DLOGBIT}(\langle p, \gamma, \alpha, 1 \rangle)$ *can be determined in polynomial time.*

Proof: Let $\langle p, \gamma, \alpha, 1 \rangle$ be a given instance of the discrete logarithm bit problem, i.e.,

- p is prime,
- γ is a primitive element of p,
- $\alpha \in \mathbb{Z}_p^*$, and
- the least significant bit of the binary representation of $\log_\gamma \alpha \bmod (p-1)$ is to be evaluated.

Computing the Parity of Discrete Logarithms by Euler

Define the function $s : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ by

$$s(w) = w^2 \bmod p.$$

Recall that

$$\mathrm{QR}_p \quad = \quad \{w^2 \bmod p \,\big|\, w \in \mathbb{Z}_p^*\}.$$

Note that $s(w) = s(p-w)$, since $p \equiv 0 \bmod p$.

Note further that

$$\begin{aligned} x^2 \equiv w^2 \bmod p \quad &\iff \quad p \text{ divides } (x-w)(x+w) \\ &\iff \quad x \equiv \pm w \bmod p. \end{aligned}$$

Computing the Parity of Discrete Logarithms by Euler

Hence, every $z \in \mathrm{QR}_p$ has exactly two preimages with respect to s.

It follows that

$$\|\mathrm{QR}_p\| = \frac{p-1}{2}.$$

In other words,

- exactly half of the elements of $\mathbb{Z}_p^*$ are quadratic residues modulo p and
- the remaining half of the elements of $\mathbb{Z}_p^*$ are quadratic nonresidues modulo p.

Since γ is a primitive element of p, $\gamma^a \in \mathrm{QR}_p$ if the exponent a is even.

Since the $(p-1)/2$ elements $\gamma^0, \gamma^2, \ldots, \gamma^{p-3}$ are pairwise distinct, they are precisely the elements of QR_p, i.e.,

$$\mathrm{QR}_p = \{\gamma^{2i} \bmod p \,\big|\, 0 \leq i \leq (p-3)/2\}.$$

Computing the Parity of Discrete Logarithms by Euler

It follows that an element α is a quadratic residue modulo p if and only if $\log_\gamma \alpha$ is even. That is,
the least significant bit of the binary representation of $\log_\gamma \alpha$ is zero
if and only if $\alpha \in \mathrm{QR}_p$,
which by Euler's criterion is equivalent to $\alpha^{(p-1)/2} \equiv 1 \bmod p$.

Hence, we have

$$\textsc{dlogbit}(\langle p, \gamma, \alpha, 1\rangle) = 0 \iff \alpha^{(p-1)/2} \equiv 1 \bmod p.$$

Since $\alpha^{(p-1)/2} \equiv 1 \bmod p$ can be efficiently computed using square-and-multiply, Euler's criterion provides an efficient algorithm for computing $\textsc{dlogbit}(\langle p, \gamma, \alpha, 1\rangle)$. ❑

What Other Bits of DLOGBIT are Easy to Compute?

Theorem

Let $\langle p, \gamma, \alpha, i \rangle$ be an instance of the discrete logarithm bit problem, and let

$$p - 1 = r2^q$$

for some odd number r. Then,

1. *for each $i \leq q$, $\text{DLOGBIT}(\langle p, \gamma, \alpha, i \rangle)$ can be determined in polynomial time, and*
2. $\log_\gamma \alpha \bmod (p-1)$ *can be computed in* $\text{FP}^{\text{DLOGBIT}(\langle p,\gamma,\alpha,q+1 \rangle)}$, *i.e., computing the $(q+1)$th bit of the discrete logarithm of α is no easier than computing the full discrete logarithm of α in $\mathbb{Z}_p^*$.*

What Other Bits of DLOGBIT are Easy to Compute?

Proof: The proof of the theorem makes use of the following lemma.

Lemma

Let p be a prime number with $p \equiv 3 \bmod 4$.

1. *Every $\alpha \in \mathrm{QR}_p$ has the two square roots*

$$\pm\alpha^{(p+1)/4} \bmod p.$$

2. *Moreover, if γ is a primitive element of p and $\alpha \neq 0$, then*

$$\mathrm{DLOGBIT}(\langle p, \gamma, \alpha, 1\rangle) \neq \mathrm{DLOGBIT}(\langle p, \gamma, p-\alpha, 1\rangle).$$

What Other Bits of DLOGBIT are Easy to Compute?

Proof of Lemma. (1) Let γ be a generator of $\mathbb{Z}_p^*$ and let $\alpha \in \mathrm{QR}_p$.

By the proof of the previous theorem, we know that

$$\alpha = \gamma^{2i}.$$

Since modulo p

$$x = \alpha^{\frac{p+1}{4}} = \gamma^{\frac{i(p-1)}{2}+i} = (-1)^i \cdot \gamma^i, \tag{7}$$

we have

$$x^2 = (-1)^{2i} \cdot \gamma^{2i} = 1 \cdot \alpha = \alpha,$$

so $\pm x = \pm\alpha^{\frac{p+1}{4}}$ are the roots of α.

(7) holds because $\left(\gamma^{\frac{p-1}{2}}\right)^2 \equiv \gamma^{p-1} \equiv 1 \bmod p$,

but $\gamma^{\frac{p-1}{2}} \not\equiv 1 \bmod p$, since γ has order $p-1$. Thus $\gamma^{\frac{p-1}{2}} \equiv -1 \bmod p$.

What Other Bits of DLOGBIT are Easy to Compute?

(2) Let $\gamma^a \equiv \alpha \bmod p$. This implies

$$\gamma^{a+\frac{p-1}{2}} \equiv \gamma^a \cdot \gamma^{\frac{p-1}{2}} \equiv -\alpha \bmod p.$$

Since $p \equiv 3 \bmod 4$, $(p-1)/2$ is odd, which proves our claim. ❑ Lemma

We will prove the theorem only for $q = 1$.

1. This is just the previous theorem saying that $\text{DLOGBIT}(\langle p, \gamma, \alpha, 1\rangle)$ can be determined in polynomial time.
2. $p - 1 = r \cdot 2^1$ for odd r means: $p \equiv 3 \bmod 4$. We give an algorithm that computes $x_{i-1}, \ldots, x_0$ with $\log_\gamma \alpha = \sum_{j=0}^{i-1} x_j 2^j$ in polynomial time using a DLogBit-2-Oracle for $\text{DLOGBIT}(\langle p, \gamma, \alpha, 2\rangle)$.

Discrete-Log-Algorithm with DLogBit-2-Oracle

Let $\alpha \equiv \gamma^a \bmod p$ for an unknown even exponent a.

By the above lemma, we either have

$$\alpha^{(p+1)/4} \equiv \gamma^{a/2} \bmod p \quad \text{or} \quad -\alpha^{(p+1)/4} \equiv \gamma^{a/2} \bmod p.$$

Which of these two possible congruences is true, can be efficiently determined as soon as we know (by querying our oracle)

$$\text{DLOGBIT}(\langle p, \gamma, \alpha, 2 \rangle)$$

by applying our previous result that the least significant bit of the discrete logarithm can be computed in polynomial time because

$$\text{DLOGBIT}(\langle p, \gamma, \alpha, 2 \rangle) = \text{DLOGBIT}(\langle p, \gamma, \gamma^{a/2}, 1 \rangle).$$

This is the intuitive idea behind the following algorithm.

Discrete-Log-Algorithm with DLogBit-2-Oracle

DISCRETE-LOG-ALGO-WITH-DLOGBIT-2-ORACLE(p, γ, α) {

(* p is prime, γ is a primitive element of $\mathbb{Z}_p^*$, and $\alpha \in \mathbb{Z}_p^*$ *)

$x_0 := \text{DLOGBIT}(\langle p, \gamma, \alpha, 1 \rangle)$; (* Using Euler's Criterion *)

$\alpha := \alpha / \gamma^{x_0} \bmod p$;

$i := 1$;

`while` $(\alpha \neq 1)$ {

$x_i := \text{DLOGBIT}(\langle p, \gamma, \alpha, 2 \rangle)$; (* Using DLogBit-2-Oracle *)

$\omega := \alpha^{(p+1)/4} \bmod p$; (* Trying the first square root of α *)

`if` $\text{DLOGBIT}(\langle p, \gamma, \omega, 1 \rangle) = x_i$ `then` $\alpha := \omega$ `else` $\alpha := p - \omega$;

$\alpha := \alpha / \gamma^{x_i} \bmod p$;

$i := i + 1$;}

`return` "$(x_{i-1}, x_{i-2}, \ldots, x_0)$" and halt;

}

Discrete-Log-Algorithm with DLogBit-2-Oracle

Breaking Cryptosystems versus Digital Signature Schemes

- When breaking a cryptosystem, a cryptanalyst usually aims at determining the private key used and the plaintext encrypted.
- When breaking a digital signature scheme, however, a cryptanalyst usually pursues a different goal, namely,

  forging signatures of signed messages.

Types of Forgery

- **Total break:** The cryptanalyst is able to determine the private key of the sender in a digital signature scheme; e.g., Bob's secret numbers b and s in the ElGamal digital signature scheme.
  Using this private key, cryptanalyst Erich can create a valid signature for any message of his choice.

- **Selective forgery:** The cryptanalyst is able to create, with nonnegligible probability of success, a valid signature for some message chosen by somebody else.
  That is, if Erich intercepts a message m that was previously not signed by Bob, he is able to create a valid signature for m with a certain success probability.

Types of Forgery

- **Existential forgery:** The cryptanalyst is able to create a valid signature for at least one message that was previously not signed by Bob. Here, no specified probability of success is required.

Again, one can distinguish several levels of security, depending on what information is available to the cryptanalyst during the attack:

- **Key-only attack:** Cryptanalyst Erich only knows Bob's public key.
- **Known-message attack:** Erich knows some pairs of messages and corresponding signatures in addition to the public key.
- **Chosen-message attack:** Erich knows the public key and obtains a list of Bob's signatures corresponding to a list of messages he has chosen at will.

Breaking the ElGamal Digital Signature Scheme

- Suppose that m is the message Erich wants to sign with a forged signature that looks like Bob's signature.

- According to the ElGamal digital signature scheme, he has to choose some elements σ and ρ satisfying

$$\begin{aligned} \sigma &= \gamma^s \bmod p \\ \rho &= (m - b\sigma)s^{-1} \bmod (p-1). \end{aligned}$$

The order in which σ and ρ are chosen does matter here.
Consider:

1. Erich choosing σ first and then the corresponding ρ;
2. Erich choosing ρ first and then the corresponding σ;
3. Erich choosing σ and ρ simultaneously;
4. a *key-only attack allowing an existential forgery*.

Breaking the ElGamal Digital Signature Scheme

1. **Erich chooses σ first and then computes the corresponding ρ:**
   By the verification condition (5):

$$\gamma^m \equiv \beta^\sigma \sigma^\rho \bmod p,$$

   in this case he must solve the discrete logarithm

$$\rho = \log_\sigma \beta^{-\sigma} \gamma^m \bmod p.$$

Breaking the ElGamal Digital Signature Scheme

2. **Erich chooses ρ first and then computes the corresponding σ:**
   He faces the problem of solving the ElGamal verification condition (5) for the unknown σ.

   This problem is not known to have an efficient algorithm either.

   However, it does not seem to be closely related to other thoroughly investigated problems such as the discrete logarithm problem.

   Thus, it might well be that there exists such an efficient solution to this problem that just eluded us so far.

   It might also be the case that there is some clever way of determining σ and ρ simultaneously so that (σ, ρ) is a valid signature for m that Alice would have to accept when verifying it using Bob's public key β.

Breaking the ElGamal Digital Signature Scheme

3. **Erich might try to choose σ and ρ simultaneously:**
   Then he must solve (5) for the unknown value m.

   In this case, he again faces the problem of computing a discrete logarithm, namely

   $$m = \log_\gamma \beta^\sigma \sigma^\rho \bmod p.$$

   This approach has the disadvantage that, depending on the choice of σ and ρ, the message signed may not be meaningful.

   Again, since solving discrete logarithms is considered to be hard, this is not a practical attack.

Key-Only Attack on ElGamal's Digital Signature Scheme

4. **Key-only attack allowing an existential forgery:**

   Erich is able to create a valid ElGamal signature for a random message m, by choosing σ, ρ, and m simultaneously.

   Let x and y be integers with $0 \leq x \leq p-2$ and $0 \leq y \leq p-2$.

   Writing σ as $\sigma = \gamma^x \beta^y \bmod p$ implies that the ElGamal verification condition (5) is of the form

$$\gamma^m \equiv \beta^\sigma (\gamma^x \beta^y)^\rho \bmod p,$$

   which is equivalent to

$$\gamma^{m-x\rho} \equiv \beta^{\sigma+y\rho} \bmod p. \tag{8}$$

Key-Only Attack on ElGamal's Digital Signature Scheme

Now, (8) is true if and only if the following two congruences are satisfied:

$$m - x\rho \equiv 0 \bmod (p-1); \tag{9}$$

$$\sigma + y\rho \equiv 0 \bmod (p-1). \tag{10}$$

Given x and y and assuming that $\gcd(y, p-1) = 1$, the congruences (9) and (10) can easily be solved for ρ and m, and we obtain:

$$\begin{aligned} \sigma &= \gamma^x \beta^y \bmod p; \\ \rho &= -\sigma y^{-1} \bmod (p-1); \\ m &= -x\sigma y^{-1} \bmod (p-1). \end{aligned}$$

By way of construction, (σ, ρ) is a valid signature for the message m.

Key-Only Attack on ElGamal's Digital Signature Scheme

Example (continuing our previous example)

- Let $p = 1367$ be a given prime number, and let $\gamma = 5$ be a given primitive element of 1367.
- Bob's private exponents are $b = 513$ and $s = 129$, which Erich does not know.
- However, Erich does know Bob's public value $\beta = 855$.
- Suppose he chooses $x = 33$ and $y = 77$.

Key-Only Attack on ElGamal's Digital Signature Scheme

Example (continued)

- Using the extended Euclidean algorithm, he checks that
  - $\gcd(77, 1366) = 1$ and
  - determines the inverse element $y^{-1} = 479$ of $y = 77$ modulo $p - 1 = 1366$.

- Then Erich computes:

$$\begin{aligned} \sigma &= 5^{33} \cdot 855^{77} \equiv 906 \cdot 343 \equiv 449 \bmod 1367; \\ \rho &= -449 \cdot 479 \equiv 757 \bmod 1366; \\ m &= -33 \cdot 449 \cdot 479 \equiv 393 \bmod 1366. \end{aligned}$$

Key-Only Attack on ElGamal's Digital Signature Scheme

Example (continued)

- Hence, $(449, 757)$ is a valid signature for the message $m = 393$.
- As a check, note that Alice will verify it using the condition (5):

$$5^{393} \equiv 1125 \equiv 930 \cdot 817 \equiv 855^{449} \cdot 449^{757} \bmod 1367$$

and will thus accept Erich's forgery.

- It is not certain, though, that 393 indeed is a message that Erich would wish to send to Alice with Bob's forged signature.

Known-Message Attack on ElGamal's Digital Signatures

This attack provides another existential forgery of ElGamal signatures:

- Suppose that Erich knows a previous signature $(\hat{\sigma}, \hat{\rho})$ for some message $\hat{m}$.
- He can then sign new messages forging Bob's signature.
  As usual, let p be a prime number with primitive element γ, and let β be Bob's public key. Let $x, y, z \in \mathbb{Z}_{p-1}$ be chosen such that

$$\gcd(x\hat{\sigma} - z\hat{\rho}, p-1) = 1.$$

- Erich computes:

$$\begin{aligned} \sigma &= \hat{\sigma}^x \gamma^y \beta^z \bmod p; \\ \rho &= \hat{\rho}\sigma(x\hat{\sigma} - z\hat{\rho})^{-1} \bmod (p-1); \\ m &= \sigma(x\hat{m} + y\hat{\rho})(x\hat{\sigma} - z\hat{\rho})^{-1} \bmod (p-1). \end{aligned} \tag{11}$$

Known-Message Attack on ElGamal's Digital Signatures

- Exercise: Check that the ElGamal verification condition (5):

$$\gamma^m \equiv \beta^\sigma \sigma^\rho \bmod p$$

  is satisfied.

- Hence, (σ, ρ) is a valid signature for the message m.

Example (Known-Message Attack on ElGamal Signatures)

As in our previous examples, Bob chooses the prime number $p = 1367$, the primitive element $\gamma = 5$ of 1367, and the public key $\beta = 855$.

Known-Message Attack on ElGamal's Digital Signatures

Example (Known-Message Attack on ElGamal Signatures: continued)

Suppose that Erich knows Bob's signature

$$(\hat{\sigma}, \hat{\rho}) = (1180, 955)$$

for the message $\hat{m} = 457$ from our previous example.

Erich chooses the integers $x = 19$, $y = 65$, and $z = 23$.

Using the extended Euclidean Algorithm, he checks that

$$\gcd(x\hat{\sigma} - z\hat{\rho}, p-1) = \gcd(455, 1366) = 1$$

and he computes the inverse element $455^{-1} \bmod 1366 = 1363$.

Known-Message Attack on ElGamal's Digital Signatures

Example (Known-Message Attack on ElGamal Signatures: continued)

Using (11), Erich now computes:

$$\begin{aligned} \sigma &= 1180^{19} \cdot 5^{65} \cdot 855^{23} \equiv 963 \cdot 674 \cdot 219 \equiv 1184 \bmod 1367; \\ \rho &= 955 \cdot 1184 \cdot 455^{-1} \equiv 984 \bmod 1366; \\ m &= 1184 \cdot (19 \cdot 457 + 65 \cdot 955) \cdot 455^{-1} \equiv 656 \bmod 1366. \end{aligned}$$

Hence, $(1184, 984)$ is a valid signature for the message 656, as can be checked by the ElGamal verification condition (5):

$$5^{656} \equiv 452 \equiv 698 \cdot 314 \equiv 855^{1184} \cdot 1184^{984} \bmod 1367.$$

Thus, Alice accepts Erich's forged signature.

Comments on the Attacks on ElGamal's Digital Signatures

Remark:

- Both of the above attacks on the ElGamal signature scheme yield an existential forgery.
- It is currently not known whether these attacks can be strengthened to yield even selective forgeries.
- Therefore, mounting these attacks is not a practical threat for the ElGamal digital signature scheme.
- As a countermeasure to prevent these attacks, one can make use of a *cryptographic hash function* as sketched below.

Countermeasure: Cryptographic Hash Functions

- In cryptography, a hashing function $h : \Sigma^* \to T$ can be used to produce a "message digest" of prespecified length from any given message of arbitrary length.
- A common choice of the length of the hash values in T is 256 bits.
- If Bob wishes to sign a message m, he first computes the message digest $t = h(m)$, which is an element of the hashing table T.
- Then, he computes his signature $s = \mathrm{sig}_B(t)$ for t, using a digital signature scheme such as ElGamal, and sends (m, s) to Alice.
- To verify the signature, she first reconstructs the message digest $t = h(m)$, using the public hashing function h, and then checks the validity of the signature s for t.

Countermeasure: Cryptographic Hash Functions

- To prevent existential forgeries by key-only or known-message attacks, hashing functions are required to have certain properties so as to be considered cryptographically secure.
- For example, suppose that Erich mounts a known-message attack.
- Thus, he already knows some pair (m, s), where
  - m is a message previously signed by Bob and
  - s is the signature for the message digest $t = h(m)$ generated from m by some hashing function h.
- Since h is public, Erich can determine t.

Countermeasure: Cryptographic Hash Functions

- He might then try to find some other mesage $\tilde{m} \neq m$ such that $h(\tilde{m}) = h(m)$.
- This would enable him to forge Bob's signature for the message $\tilde{m}$, since the pair $(\tilde{m}, s)$ contains a valid signature s for $\tilde{m}$.
- This type of attack can be prevented by requiring the hashing function used to be "collision-free" on the relevant domain in the sense that it is computationally infeasible to determine, given m, some message $\tilde{m}$ with

$$\tilde{m} \neq m \quad \text{and} \quad h(\tilde{m}) = h(m).$$

Careful Choice of ElGamal Parameters Avoids Total Break

Some care must be taken in choosing the parameters of the ElGamal Digital Signature Scheme in order to avoid a total break:

1. **Bob's secret exponent s must never be revealed.**
   - If Erich knows s, then it is a matter of routine for him to compute, using (4), Bob's secret exponent b from m and the signature (σ, ρ) by

$$b \equiv (m - s\rho)\sigma^{-1} \bmod p - 1.$$

   - This known-message attack results in a total break of the ElGamal digital signature scheme, and Erich can henceforth forge Bob's signature at will.

Careful Choice of ElGamal Parameters Avoids Total Break

2. **Bob's secret exponent s must never be used twice for signing distinct messages.**
   - Again, a known-message attack can be mounted to yield a total break.
   - Bob computes

$$\beta = \gamma^b \bmod p \quad \text{with his private } b,$$
$$\sigma = \gamma^s \bmod p \quad \text{with his private } s,$$

but uses s twice for messages m_1 and m_2, $m_1 \neq m_2$:

$$\rho_1 = (m_1 - b\sigma)s^{-1} \bmod (p-1),$$
$$\rho_2 = (m_2 - b\sigma)s^{-1} \bmod (p-1),$$

yielding the signatures (σ, ρ_1) for m_1 and (σ, ρ_2) for m_2.

Careful Choice of ElGamal Parameters Avoids Total Break

- Alice can now verify the signatures for $i \in \{1,2\}$:

$$\gamma^{m_i} \equiv \beta^{\sigma} \sigma^{\rho_i} \bmod p. \tag{12}$$

- Erich knows p, γ, β, and (σ, ρ_i) for m_i, $i \in \{1,2\}$.
  That is, this is a known-message attack.

- From (12) it follows that

$$\gamma^{m_1 - m_2} \equiv \beta^{\sigma - \sigma} \sigma^{\rho_1 - \rho_2} \equiv \gamma^{s(\rho_1 - \rho_2)} \bmod p,$$

  which is equivalent to:

$$m_1 - m_2 \equiv s(\rho_1 - \rho_2) \bmod (p-1). \tag{13}$$

Careful Choice of ElGamal Parameters Avoids Total Break

- Let $d = \gcd(\rho_1 - \rho_2, p-1)$.

  Since d divides $\rho_1 - \rho_2$ and $p-1$, d also divides $m_1 - m_2$.

- Erich first determines d and then

$$m' = \frac{m_1 - m_2}{d}, \quad \rho' = \frac{\rho_1 - \rho_2}{d}, \quad \text{and} \quad p' = \frac{p-1}{d}.$$

- By (13), it follows that

$$m' \equiv s \cdot \rho' \bmod p'.$$

- Since $\gcd(\rho', p') = 1$, we have $s = m' \cdot \rho'^{-1} \bmod p'$.

Careful Choice of ElGamal Parameters Avoids Total Break

- Specifically, there are d candidate values for s:

$$s_i = m' \cdot \rho'^{-1} + i \cdot p' \bmod (p-1) \quad \text{for } 0 \leq i \leq d-1.$$

  The correct value can be determined via $\sigma = \gamma^s \bmod p$.

- From, say $\rho_1 = (m_1 - b\sigma)s^{-1} \bmod (p-1)$, Erich can now easily determine

$$b = (m_1 - s\rho_1)\sigma^{-1} \bmod (p-1).$$

- $\Rightarrow$ **total break!**