NTNU | Norwegian University of Science and Technology

PROTOCOL COMPOSITION 1

TTM4205 – Lecture 15

Jonathan Komada Eriksen

25.10.2023

Contents

Contents

OpenPGP

- Recall TTM4135: Securing email.
- Standardised in RFC4880
- Encryption: ElGamal Hybrid Encryption (...or RSA).
- Signatures: DSA or RSA.
- *Today:* Cross-implementation attack on OpenPGP.

On the (in)security of ElGamal in OpenPGP

Luca De Feo*
IBM Research Europe – Zurich
Rüschlikon, Switzerland

Bertram Poettering*
IBM Research Europe – Zurich
Rüschlikon, Switzerland

Alessandro Sorniotti*
IBM Research Europe – Zurich
Rüschlikon, Switzerland

Figure: On the (in)security of ElGamal in OpenPGP

ElGamal Hybrid Encryption

Key Generation

- Work in the group $G = \langle g \rangle$.
- Secret key: $\mathsf{sk} = x$.
- Public key: $\mathsf{pk} = X = g^x$

ElGamal Hybrid Encryption

Key Generation

- Work in the group $G = \langle g \rangle$.
- Secret key: $\mathsf{sk} = x$.
- Public key: $\mathsf{pk} = X = g^x$

Encryption

- Select y, compute $Y = g^y$, and $Z = X^y = g^{xy}$.
- Use Z as a symmetric key, to encrypt message m to ct.
- Send $C = (Y, \mathrm{ct})$

ElGamal Hybrid Encryption

Key Generation

- Work in the group $G = \langle g \rangle$.
- Secret key: $\mathsf{sk} = x$.
- Public key: $\mathsf{pk} = X = g^x$

Encryption

- Select y, compute $Y = g^y$, and $Z = X^y = g^{xy}$.
- Use Z as a symmetric key, to encrypt message m to ct.
- Send $C = (Y, \mathrm{ct})$

Decryption

- Compute $Z = Y^x$ and use Z to decrypt ct

ElGamal Hybrid Encryption

Key Generation

- Work in the group G, select some generator $g \in G$.
- Secret key: $\mathsf{sk} = x$.
- Public key: $\mathsf{pk} = X = g^x$

Questions

What group should G be? How should g be selected?? What interval should x and y be picked from??? We'll see four different configurations that are *all used in practice*. In all cases, $G = (\mathbb{Z}/p\mathbb{Z})^\times$ for some prime p.

Repitition?

- Recall that $(\mathbb{Z}/p\mathbb{Z})^\times$ is *cyclic* of *order* $p-1$.
- Let $p-1 = q_1 q_2 \cdots q_n$, with q_i relatively prime powers.
- $(\mathbb{Z}/p\mathbb{Z})^\times \cong \mathbb{Z}/(p-1)\mathbb{Z} \cong \mathbb{Z}/q_1\mathbb{Z} \times \mathbb{Z}/q_2\mathbb{Z} \times \cdots \times \mathbb{Z}/q_n\mathbb{Z}$.

Two easy configs we'll focus on

Configuration A

- $G = (\mathbb{Z}/p\mathbb{Z})^\times$ where $p-1$ has at least one large prime factor.
- g should be a generator of G.
- x, y picked from $[0, p-1]$.

Configuration B

- $G = (\mathbb{Z}/p\mathbb{Z})^\times$ where $p-1$ has at least one large prime factor, say q.
- g should be a generator of the subgroup $G' \subseteq G$, of order q
- x, y should be picked from $[0, q-1]$ for efficiency.

Note that in Configuration B, $q \ll p$.

Two more

Configuration C - Safe Primes

- $G = (\mathbb{Z}/p\mathbb{Z})^\times$ where $p - 1 = 2q$, where q is prime.
- $g = 4$ (note that this is a generator of the group $G' \subseteq G$ of order q)
- x, y picked from $[0, p - 1]$.

Configuration C - Lim-Lee Primes

- $G' = (\mathbb{Z}/p\mathbb{Z})^\times$ where $p - 1 = 2q_1 q_2 \cdot q_n$, with q_i all different primes of roughly the same size.
- g should be a generator of the subgroup $G' \subseteq G$, of order q_i for some i.
- x, y should be picked from $[0, q_i - 1]$ for efficiency.

Contents

Pohlig-Hellman

Discrete logartihm

Let $G = \langle g \rangle$, with $|G| = p_1^{e_1} p_2^{e_2} \cdots p_n^{e_n}$, where p_i are prime powers. Given $X \in G$, compute x s.t. $g^x = X$.

Pohlig-Hellman

Discrete logartihm

Let $G = \langle g \rangle$, with $|G| = p_1^{e_1} p_2^{e_2} \cdots p_n^{e_n}$, where p_i are prime powers. Given $X \in G$, compute x s.t. $g^x = X$.

- Pohlig-Hellman reduces this to the task of computing discrete logs in groups of order p_i.

Pohlig-Hellman

Discrete logartihm

Let $G = \langle g \rangle$, with $|G| = p_1^{e_1} p_2^{e_2} \cdots p_n^{e_n}$, where p_i are prime powers. Given $X \in G$, compute x s.t. $g^x = X$.

- Pohlig-Hellman reduces this to the task of computing discrete logs in groups of order p_i.
- Solving discrete logs in groups of prime power order.

Pohlig-Hellman

Discrete logartihm

Let $G = \langle g \rangle$, with $|G| = p_1^{e_1} p_2^{e_2} \cdots p_n^{e_n}$, where p_i are prime powers. Given $X \in G$, compute x s.t. $g^x = X$.

- Pohlig-Hellman reduces this to the task of computing discrete logs in groups of order p_i.
- Solving discrete logs in groups of prime power order.
- Combining results using CRT

Pohlig-Hellman - Prime power case

We compute the discrete log of X to base g, where g generates a group of order p^e.

Pohlig-Hellman - Prime power case

We compute the discrete log of X to base g, where g generates a group of order p^e.

1. Let $x_0 := 0$
2. Set $g_{\texttt{small}} := g^{p^{e-1}}$.
3. For $0 \leq k < e$:
   - 3.1 Compute $X_k := (g^{-x_k} X)^{p^{e-1-k}}$
   - 3.2 Compute d_k s.t. $X_k = g_{\texttt{small}}^{d_k}$
   - 3.3 Set $x_{k+1} := x_k + p^k d_k$
4. Return x_e.

To see that this algorithm is correct, write x in base p.

Pohlig-Hellman - Full algorithm

We compute the discrete log of X to base g, where g generates a group of order $|G| = N = p_1^{e_1} p_2^{e_2} \cdots p_n^{e_n}$.

Pohlig-Hellman - Full algorithm

We compute the discrete log of X to base g, where g generates a group of order $|G| = N = p_1^{e_1} p_2^{e_2} \cdots p_n^{e_n}$.

Abstractly: Since $G \simeq \mathbb{Z}/p_1^{e_1}\mathbb{Z} \times \mathbb{Z}/p_2^{e_2}\mathbb{Z} \times \cdots \times \mathbb{Z}/p_n^{e_n}\mathbb{Z}$, simply project onto each summand, and recover x with CRT.

Pohlig-Hellman - Full algorithm

We compute the discrete log of X to base g, where g generates a group of order $|G| = N = p_1^{e_1} p_2^{e_2} \cdots p_n^{e_n}$.

Abstractly: Since $G \simeq \mathbb{Z}/p_1^{e_1}\mathbb{Z} \times \mathbb{Z}/p_2^{e_2}\mathbb{Z} \times \cdots \times \mathbb{Z}/p_n^{e_n}\mathbb{Z}$, simply project onto each summand, and recover x with CRT.

Concrete:

1. For each $0 < i \leq n$:
   - 1.1 Compute $g_i := g^{N/(p_i^{e_i})}$ and $X_i = X^{N/(p_i^{e_i})}$
   - 1.2 Compute x_i s.t. $X_i = g_i^{x_i}$ using the previous algorithm.
2. We now have a system of congruences

$$x \equiv x_1 \pmod{p_1^{e_1}},$$
$$\vdots$$
$$x \equiv x_n \pmod{p_n^{e_n}},$$

which you learned to solve in kindergarden.

Reference Group

Any comments to the reference group? How’s the course going? How’s the workload? Any comments to lectures and/or exercise classes? 10 mins discussion :)

Baby Step - Giant Step

We compute the discrete log X to base g, where g generates a group of prime order p.

BS-GS

Solves in $O(\sqrt{p})$ time and memory.

Idea: Write $x = am + b$ for $m = \lceil \sqrt{p} \rceil$. Store all g^b for $b < m$, and solve for a such that $g^b = X(g^{-m})^a$.

BS-GS Algorithm

1. Set $m = \lceil \sqrt{p} \rceil$.
2. For each $0 \leq b < m$:
   2.1 Compute and save the pair (b, g^b) in a table.
3. compute $Y = g^{-m}$.
4. For each $0 \leq a < m$:
   4.1 Compute and check if XY^a is in the table, say for b.
   4.2 If so, return $am + b$.

Small note on a different algorithm

Pollard Rho

A different algorithm for the same problem as BS-GS, but which only uses constant memory. Flavor is more similar to the following algortihm....

Pollard's Kangaroo

We compute the discrete log X to base g in G, where we know that the solution x lies in some interval $[a, b]$.

Pollard's Kangaroo/Lambda Algorithm

Solves (probabilistically) in $O(\sqrt{b-a})$ time.

Requires a hash function $H : G \to S$, where S is a set of random integers, roughly of size $\sqrt{b-a}$.

Pollard's Kangaroo

1. Set $Y_0 := g^b$.
2. Set $d := 0$
3. For $0 \leq i < N$ for some bound N:
   - **3.1** Compute $Y_{i+1} = Y_i g^{f(Y_i)}$.
   - **3.2** Update $d := d + f(Y_i)$.
4. $d' := 0$
5. $X_0 := X$
   - **5.1** Compute $X_{i+1} = X_i g^{f(X_i)}$
   - **5.2** Update $d' := d' + f(X_i)$.
   - **5.3** If $X_{i+1} = Y_N$, return the solution.
   - **5.4** If $d' > b - a + d$, restart with a new choice of f.

Solution given as $X g^{d'} = X_{i+1} = Y_N = g^{b+d} \Rightarrow X = g^{b+d-d'}$

Contents

Weakness across implementations in ElGamal

- By themselves, configuration A/B/C/D are all secure.

Weakness across implementations in ElGamal

- By themselves, configuration A/B/C/D are all secure.
- However, by combining them, they can become insecure.

Weakness across implementations in ElGamal

- By themselves, configuration A/B/C/D are all secure.
- However, by combining them, they can become insecure.
- Specific attack: User using config B/D sends person using configuration A an PGP encrypted email.

Weakness across implementations in ElGamal

- By themselves, configuration A/B/C/D are all secure.
- However, by combining them, they can become insecure.
- Specific attack: User using config B/D sends person using configuration A an PGP encrypted email.
  - Sender uses small exponents $x \in [0, \ldots, 2^{256}]$.
  - Receiver uses g generator of group of order $N = p_1^{e_1} q_2^{e_2} \ldots p_n^{e_n} N'$, where p_i are all small enough primes to solve discrete logs in.

Attack

We are computing the discrete log of X to base g where:

- g generates a group of order $N = p_1^{e_1} q_2^{e_2} \dots p_n^{e_n} N'$, where p_i are all small-ish primes. Let $M := N/N'$.
- The solution x lies in $[0, \dots, 2^{256}]$

Attack

We are computing the discrete log of X to base g where:

- g generates a group of order $N = p_1^{e_1} q_2^{e_2} \dots p_n^{e_n} N'$, where p_i are all small-ish primes. Let $M := N/N'$.
- The solution x lies in $[0, \dots, 2^{256}]$

Attack:

1. Use Pohlig-Hellman combined with BS-GS (or Pollard rho) to compute $w \equiv x \pmod{M}$, by computing the dlog of $X^{N'}$ to the base $g^{N'}$.
2. Note now that $X = g^{zM+w}$ for some unknown $z \in [0, \dots, M/(2^{256})]$. Therefore, find z by using computing the discrete log of X/g^w to the base g^M, using Pollard's kangaroo.

Practicality of attack

Set computational power to 2^{50} operations.

1. To solve, we need $p-1$ to be divisible by enough small primes $p_i < 2^{100}$.
2. Same as before, write $p-1 = p_1^{e_1} q_2^{e_2} \dots p_n^{e_n} N'$, where $p_i < 2^{100}$, and let $M := N/N'$.
3. For the last step we need $(2^{256})/M < 2^{100}$.

Computing the exact probability of this happening when p and g comes from configuration A is complicated, but it happens very frequently.

Further reading

- When additionally considering side-channel attacks, the previous attack becomes even more prominent.
  - See Chp 5 in On the (in)security of ElGamal in OpenPGP

Further reading

- When additionally considering side-channel attacks, the previous attack becomes even more prominent.
  - See Chp 5 in On the (in)security of ElGamal in OpenPGP
- Bridge between this week and next week:
  - What can the attacker do when having write access to the public/encrypted private keys?
    - Why is this attack scenario realistic? Cloud based key management etc.
  - Turns out, quite a lot

Victory by KO: Attacking OpenPGP Using Key Overwriting

Lara Bruseghini*
ETH Zurich
Switzerland
larabr@protonmail.com

Daniel Huigens
Proton AG
Switzerland
d.huigens@protonmail.com

Kenneth G. Paterson
ETH Zurich
Switzerland
kenny.paterson@inf.ethz.ch

Figure: Victory by KO: Attacking OpenPGP Using Key Overwriting

Questions?