SECURITY OF THE CAO-LI PUBLIC KEY CRYPTOSYSTEM

LEK-HENG LIM

Abstract. We show that the Cao-Li cryptosystem proposed in [1] is not secure. Its private key can be reconstructed from its public key using elementary means such as LU-decomposition and Euclidean algorithm.

1. Description of the Cryptosystem

The Cao-Li public key cryptosystem was first proposed in [1]. It encrypts messages using a bilinear form that is chosen to permit easy decryption by the Chinese remainder theorem. Public key cryptosystems that are designed along this line are not uncommon in the Chinese cryptographic literature. However, as most of the original papers were published in Chinese, they remained relatively obscure until a few of them were described in [2] (in English) recently. Our description below is based on the latter reference.

Let $p_1, \dots, p_n$ be n distinct primes where $p_i \equiv 3 \pmod 4$. For $i = 1, \dots, n$, define

$$m_i := \frac{1}{p_i}\left(\prod_{j=1}^{n} p_j\right).$$

Compute for each m_i, an integer m_i' that satisfies $m_i' m_i \equiv 1 \pmod{p_i}$ and $0 < m_i' < p_i$. We define positive integers

$$\lambda_i := m_i' m_i$$

for $i = 1, \dots, n$ and the diagonal matrix

$$\Lambda := \operatorname{diag}[\lambda_1, \dots, \lambda_n].$$

Note that

$$\lambda_i \equiv \delta_{ij} \pmod{p_j} \tag{1}$$

where δ_{ij} is 1 if $i = j$ and 0 otherwise.

We choose another two invertible $n \times n$ lower-triangular matrices P_1 and P_2 with non-negative integer entries that are bounded by

$$\beta := \min_{1 \le i \le n} \sqrt{\frac{p_i}{i(i+1)d}} \tag{2}$$

where $d \geq 1$ is a chosen positive integer.

The secret key comprises the two matrices P_1, P_2 and the primes p_i, $i = 1, \dots, n$. The public key is the $n \times n$ symmetric matrix B given by

$$B := P_2^T P_1^T \Lambda P_1 P_2.$$

Let the message block be $\mathbf{x} = (x_1, \ldots, x_n)$ where $0 \le x_i \le d$. The ciphertext y is computed as

$$y = \mathbf{x}B\mathbf{x}^T.$$

If we let $\mathbf{z} := \mathbf{x}P_2^T P_1^T$, then

$$y = \mathbf{z}\Lambda\mathbf{z}^T = \lambda_1 z_1^2 + \cdots + \lambda_n z_n^2.$$

From (1), we have

$$z_k^2 \equiv y \pmod{p_k}. \tag{3}$$

Keeping in mind that P_1^T and P_2^T are upper-triangular and their entries are non-negative and bounded by β, we have, from (2) and $0 \le x_i \le d$, that

$$0 \le z_k \le \sum_{i=1}^{k} \sum_{j=i}^{k} d\beta^2 = d\beta^2 \frac{k(k+1)}{2} < \frac{p_k}{2}. \tag{4}$$

We can carry out decryption as follows. For each $k = 1, \ldots, n$, compute the unique z_k satisfying (3) and (4). The message can then be recovered by

$$\mathbf{x} = \mathbf{z}\left(P_2^T P_1^T\right)^{-1}. \tag{5}$$

Note that since $p_k \equiv 3 \pmod 4$, effective algorithms for computing square roots $\pmod{p_k}$ exist (see [3]).

2. Key Recovery

We will first recover Λ from B. Let $P_1P_2 =: P = (p_{ij})_{1\le i,j\le n}$. Then P is an invertible lower-triangular matrix with non-negative integral entries by the same properties of P_1 and P_2. Since P is invertible and has non-negative integral entries, we have $\det P = 1$. Moreover, we also have $\det P = p_{11} \times \cdots \times p_{nn}$ since P is triangular. As all the p_{ii}'s are non-negative, it then follows that $p_{ii} = 1$ for $i = 1, \ldots, n$.

Λ and P can be recovered from B using an algorithm that is very similar to the algorithm for LU-decomposition of a matrix (the difference being that row reduction is done starting from the bottom rows). Denote the ith row of B by $\mathbf{b}_i = (b_{i1}, \ldots, b_{in})$, $i = 1, \ldots, n$. We know immediately that $b_{nn} = \lambda_n$.

Algorithm A	
Input.	$B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)^T = (b_{ij})_{1\le i,j\le n}$
Output.	$\lambda_1, \ldots, \lambda_n, P$

Step 1.	for $i = n-1, n-2, \ldots, 1$ do
	    for $j = n, n-1, \ldots, i+1$ do
	        $\mathbf{b}_i \leftarrow \mathbf{b}_i - \frac{b_{ji}}{b_{jj}}\mathbf{b}_j$;
	    end;
	end;

Step 2.	for $i = 1, \ldots, n$ do
	    $\lambda_i \leftarrow b_{ii}$;
	    $\mathbf{b}_i \leftarrow \mathbf{b}_i/\lambda_i$;
	end;
	$P \leftarrow B$;

The following shows that Algorithm A indeed yields the required output. Let the ith row of P be $\mathbf{p}_i$, $i = 1, \ldots, n$. Since $p_{ji} = 0$ if $j < i$ and $p_{ii} = 1$, we may write $\mathbf{b}_i = \lambda_i \mathbf{p}_i + \sum_{j=i+1}^{n} \lambda_j p_{ji} \mathbf{p}_j$. For each $i = n-1, n-2, \ldots, 1$, the inner loop of Step 1 effectively does

$$\mathbf{b}_i \leftarrow \mathbf{b}_i - \sum_{j=i+1}^{n} \frac{b_{ji}}{b_{jj}} \mathbf{b}_j.$$

We shall show inductively that $\mathbf{b}_i$ is reduced to $\lambda_i \mathbf{p}_i$ at stage i: clearly $\mathbf{b}_n = \lambda_n \mathbf{p}_n$; suppose $\mathbf{b}_i$ is reduced to $\lambda_i \mathbf{p}_i$ at stage $i = n-1, \ldots, n-k$, then at stage $n-k-1$,

$$\begin{aligned} \mathbf{b}_{n-k-1} \leftarrow \mathbf{b}_{n-k-1} - \sum_{j=n-k}^{n} \frac{b_{ji}}{b_{jj}} \mathbf{b}_j \\ = \mathbf{b}_{n-k-1} - \sum_{j=n-k}^{n} \lambda_j p_{ji} \mathbf{p}_j \\ = \lambda_{n-k-1} \mathbf{p}_{n-k-1}. \end{aligned}$$

Hence Step 1 reduces $B = (\mathbf{b}_1, \ldots, \mathbf{b}_n)^T$ to $(\lambda_n \mathbf{p}_n, \ldots, \lambda_n \mathbf{p}_n)^T = \Lambda P$. Since the diagonal entries of P are all 1's, the diagonal entries of ΛP are the required λ_i's. Consequently, P can be recovered by dividing each row by its corresponding diagonal entry.

We can now recover the moduli $p_1, \ldots, p_n$ from $\lambda_1, \ldots, \lambda_n$. From (1), we see that for a fixed i, $p_i \mid \lambda_j$ for all $j \neq i$ and $p_i \mid \lambda_i - 1$. So

$$p_i \mid d_i := \gcd(\lambda_1, \ldots, \lambda_{i-1}, \lambda_i - 1, \lambda_{i+1}, \ldots, \lambda_n).$$

It could of course happen that $d_i \neq p_i$ for some i. So this process only partially recovers the p_i's. However our computer simulations (using C++ with LiDIA) show that instances where $d_i \neq p_i$ are rare. We shall give some heuristics to substantiate this claim. For $d_i = p_i$, it is sufficient that $\gcd(m'_1, \ldots, m'_{i-1}, m'_{i+1}, \ldots, m'_n) = 1$. From [4], we have

$$\begin{aligned} \#\big\{(a_1, \ldots, a_k) \in \mathbb{N}^k \;\big|\; \gcd(a_1, \ldots, a_k) = 1, \text{ all } a_i \leq N\big\} \\ = \begin{cases} N^k/\zeta(k) + \mathrm{O}(N^{k-1}) & \text{if } k > 2, \\ 6N^2/\pi^2 + \mathrm{O}(N \log N) & \text{if } k = 2. \end{cases} \end{aligned}$$

where $\zeta(s) = \sum_{i=1}^{\infty} i^{-s}$ is the Riemann zeta function. Assuming that each m'_i is randomly distributed in $\{1, \ldots, N\}$ where $N := \max\{p_1, \ldots, p_n\}$, the probability that $\gcd(m'_1, \ldots, m'_{i-1}, m'_{i+1}, \ldots, m'_n) = 1$ is then at least $\zeta(n-1) \geq 6/\pi^2 \approx 0.60$ when N is large enough. So we can expect to recover more than half of the p_i's. In fact our simulations show that we almost always have $d_i = p_i$ and many of the rare exceptions are of the form $d_i = 2p_i$ where p_i can also be recovered easily.

3. Conclusion

Note that Algorithm A is essentially LU-decomposition and the d_i's can be computed using the Euclidean algorithm. Since these two methods can be carried out efficiently, we can easily recover P and most of the p_i's. It then follows that the Cao-Li cryptosystem is insecure and thus should not be used.

References

[1] Z.F. Cao and Y.C. Li, "A Matrix-Covering Public-Key Cryptosystem," (in Chinese) Research Report of the Harbin University of Industry, pp. 1–37, 1991.

[2] C. Ding, D. Pei and A. Salomaa, *Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography*, World Scientific, Singapore, 1996.

[3] H. Cohen, *A Course in Computational Algebraic Number Theory*, Springer-Verlag, Berlin Heidelberg, 1993.

[4] E. Hlawka, J. Schoiβengeier and R. Taschner, *Geometric and Analytic Number Theory*, Springer-Verlag, Berlin Heidelberg, 1991.

DSO National Laboratories, 25th Storey, Tower A, Defence Technology Towers, Depot Road, Singapore 109679

Current address: Department of Mathematics, Malott Hall, Cornell University, Ithaca, NY, 14853-4201

E-mail address: `lekheng@math.cornell.edu`