Reaction Attacks Against Various Public-Key Cryptosystems

J. David Mitchell
Western University

Introduction

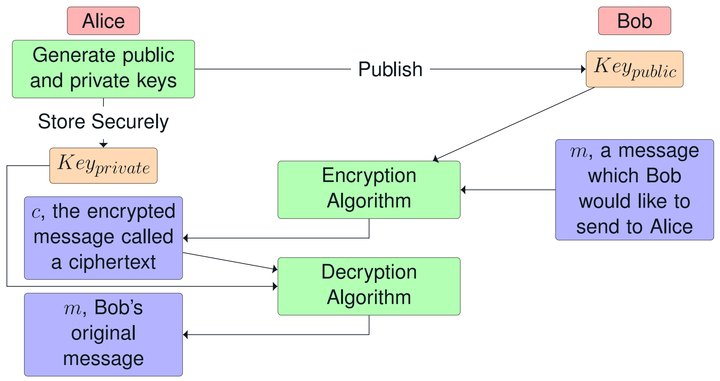
Fundamental to the design of a secure public-key cryptosystem (PKC) is the discovery of a difficult to solve problem and an associated solution which is easy to verify. In practice these problems are hard to find, and it is even more difficult to prove their difficulty. This forces system designers to rely on a plausible assumption of difficulty. However, developments in quantum computing, such as Shor's algorithm [5], have challenged these assumptions in existing popular systems. This has led to a search for systems based on problems which remain difficult in a post-quantum world. An important part of evaluating these systems is to ensure that they also remain secure against traditional (non-quantum) attacks. This poster outlines two systems based on problems that are currently thought to be hard, even for quantum computers, and traditional attacks on these systems, known as *Reaction Attacks*, introduced by Hall, Goldberg and Schneier [3].

Public-Key Cryptography

Suppose that Bob would like to send Alice a secret message. One way to accomplish this is to have Alice set up a public lock-box with a one-way slot for accepting letters and only one key, which she owns. Bob can then send his secret message by putting it into the box. He knows that Alice will be the only one who can read it as she owns the only key to the box. Alice and Bob's system captures the essence of a PKC. The idea of a box which cannot be opened easily by anyone but Alice, since she owns the only key, captures the concept of how the hard problems with easy to verify solutions described in the introduction are used.
A PKC can be separated into three parts:

- **Key Generation:** Choose a private and public key, these are analogous to Alice's lock-box key and lock-box, respectively.
- **Encryption Algorithm:** Analogous to placing a message in the lock-box.
- **Decryption Algorithm:** Analogous to opening the lock-box with the key.

The following diagram outlines how these parts come together to form the PKC:



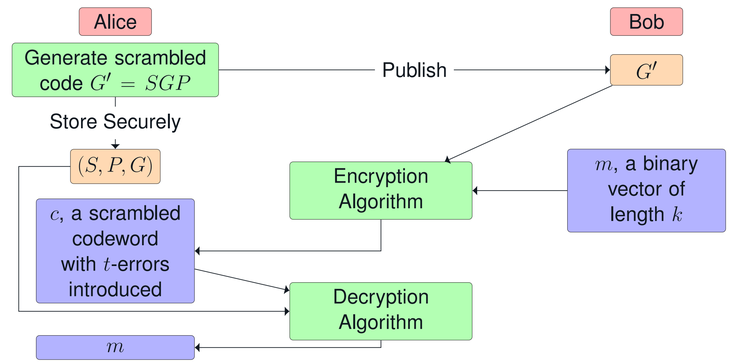
Reaction Attack

As in the introduction, one way to break a PKC is to attack the difficulty of its underlying hard problem. However, this is not the only way a system can be attacked. A reaction attack circumvents the system by taking advantage of an "information leak" when observing Alice's reaction to a decrypted message. In our lock-box analogy, suppose that Eve can discern information about Bob's message by watching how Alice reacts after reading his message. Eve can then attempt to recover Bob's message from an intercepted ciphertext by sending specific messages to Alice and observing her reactions.

McEliece Cryptosystem

In 1978, McEliece [4] introduced a new PKC based on the difficulty of decoding arbitrary linear codes. A linear code can be thought of as an encoder which turns messages into codewords and a decoder capable of removing a limited number of errors from a codeword before returning a message. McEliece's cryptosystem is outlined as follows.

- **Key Generation:**
  1. Choose G a $k \times n$ generator matrix for a t-error correcting binary linear code, S a random $k \times k$ invertible matrix and P a random $n \times n$ permutation matrix.
  2. Publish $Key_{public} = G' = SGP$ and securely store $Key_{private} = (S, P, G)$.
- **Encryption Algorithm:** For the message m, a binary vector of length k, compute the ciphertext $c = mG' + e$, where e is a random binary vector of length n and weight t. The ciphertext is a "scrambled" codeword with t errors added.
- **Decryption Algorithm:**
  1. Compute $c' = cP^{-1}$, a codeword of the linear code G with errors.
  2. Remove errors from c' using the $t-$error correcting decoder to get $m' = mS$.
  3. Compute $m = m'S^{-1}$.



Reaction Attack Against McEliece

Suppose Eve intercepts c, a ciphertext which Bob intended to send to Alice. Eve can retrieve Bob's original message m from c using the following algorithms:

Algorithm A: Compute c', a ciphertext with exactly $t+1$ errors.

1. Let $i = 1$.
2. Let c' equal c with bits 1 through i flipped and determine if c' decodes correctly by sending it to Alice and observing her reaction.
3. If so, then increase i by 1 and repeat step 2. Otherwise return c'.

Algorithm B: Locate and remove errors in c'.

1. One bit at a time, flip each bit of c' to form c'' and determine if c'' decodes correctly as in Algorithm A.
2. If so, record the bit as an error. If all bits have been flipped remove the recorded errors from c' and return c''', an error-free scrambled codeword. Otherwise continue to the next bit.

Algorithm C: Compute Bob's message m.

1. Choose k bits of c''' such that G'_k, the $k \times k$ matrix formed by the corresponding columns of G', is invertible.
2. Compute $m = (c_k)G'^{-1}_k$, where c_k is the vector formed by the k bits of c'''.

Modified Ajtai-Dwork Cryptosystem

In 1997, Goldreich, Goldwasser, and Halevi [2] presented a modified version of a cryptosystem from Ajtai and Dwork [1] based on the assumed difficulty of the hidden hyperplane problem. This modified system is outlined below.

- **Key Generation:**
  1. Specify a security parameter n and securely store $Key_{private} = u$, a random vector in the open unit ball of $\mathbb{R}^n$.
  2. Use u to choose two lists of vectors, denoted $\mathbf{v} = (v_1, ..., v_r)$ and $\mathbf{w} = (w_1, ..., w_n)$, and an index $i_1 \in \{1, \dots r\}$. Publish $Key_{public} = (\mathbf{v}, \mathbf{w}, i_1)$.
- **Encryption Algorithm:** For a binary vector m each bit m_j is encrypted as a vector by choosing random $b_1, \dots, b_r \in \{0, 1\}$ and reducing the vector $c_j = \frac{1}{2}v_{i_1}m_j + \sum_i b_i v_i$ modulo $P(\mathbf{w})$, the fundamental parallelepiped of $\mathbf{w}$.
- **Decryption Algorithm:** For a ciphertext $\mathbf{c} = (c_1, \dots, c_j)$ each c_i is decrypted as 0 if $\langle u, c_i \rangle$ is within $1/4$ of an integer and 1 otherwise.

Reaction Attack Against Modified Ajtai-Dwork

Assume there exists an oracle $O(v)$ which returns the plaintext obtained when decrypting the single vector v. Eve can use this oracle and the following algorithms to retrieve Alice's private key $u = (u_1, \dots, u_n)$:

Algorithm D: Determine the binary expansion of $|u_i|$ for each i.

1. Specify r and let $|u_i| = d_0.d_1, \dots d_r$ be the binary expansion with $d_0 = 0$.
2. For each $j \geq 0$ compute $d_{j+1} = O(v) \oplus d_j$, where v is the vector with 2^{j-1} as the i^{th} component and 0 everywhere else.

Algorithm E: Determine the sign of u_i for each i.

1. Assume $u_1 > 0$. For each $j > 1$ compute $u_1 + |u_j|$, $u_1 - |u_j|$ and determine the first bit k in which $u_1 + |u_j|$ and $u_1 - |u_j|$ differ.
2. Determine the k^{th} bit of $u_1 + u_j$, and therefore the sign of u_j, by computing $O(v)$ where v is the vector with 2^{k-3} in the 1st and j^{th} components and zero elsewhere.

Acknowledgements

I would like to extend my gratitude to Dr. Chris Hall for supervising my research internship and the Western Undergraduate Research Internship (USRI) Program for funding the internship. I would also like to extend my thanks to Western Libraries, Western Student Experience and Western Research for supporting the USRI program.

References

[1] M. Ajtai and C. Dwork. "A Public-Key Cryptosystem with Worst-Case/Average-Case Equivalence". In: *Proceedings of the twenty-ninth annual ACM symposium on theory of computing* (1997), pp. 284–293.

[2] O. Goldreich, S. Goldwasser, and S. Halevi. "Eliminating Decryption Errors in the Ajtai-Dwork Cryptosystem". In: *Advances in Cryptology — CRYPTO '97* (2006), pp. 105–111.

[3] C. Hall, I. Goldberg, and B. Schneier. "Reaction Attacks against Several Public-Key Cryptosystems". In: *Lecture Notes in Computer Science* 1726 (1999), pp. 2–12.

[4] R.J. McEliece. "A Public-Key Cryptosystem Based on Algebraic Coding Theory". In: *Deep Space Network Progress Report 42-44* (1978), pp. 114–116.

[5] P.W. Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM Journal on Computing* 26 (1997), pp. 1484–1509.