Introduction

In this paper, we will investigate the state of contemporary cryptography through several different approaches and implementations. Within each approach and implementation, we will describe their most common cryptographic protocols. Through analysis of these protocols' efficiency and security, we can conclude the current best-practices in cryptography for obtaining optimal results.

Cryptography describes the practice of applying an encoding protocol to data such that the resulting encoded data bears no obvious semblance to the original data, yet, when the proper decoding protocol is applied to the data, the original data is returned. Each protocol takes a key as its parameter, which can be a public, private, or secret key. A private key is assumed to be known only by the entity which generated it, a secret key assumed to be known by two or more entities, and a public key is assumed to be known by any entity.

The objective function of a cryptosystem (composed of an encoding and decoding algorithm) is to secure data as well as possible. In order to asses this goal, we will only be analyzing the computational difficulty of breaking a cryptosystem. A cryptosystem is considered broken when the private or secret key(s) can be derived from public information (the encrypted data, the encryption scheme, and any public keys) in a computationally feasible amount of time.

Paradigms of cryptography

The primary two cryptography approaches are symmetric-key encryption and public-key cryptography. Symmetric-key encryption denotes a system whose encrypting and decrypting keys can easily be computed by knowing only one of them [8, p. 18]. Hence, only the communicating parties should know the one (or more) secret keys. Public-key cryptography utilizes a private key, generated by a single entity, and public key, generated from the private key. Any data encrypted with a public key can be only be decrypted with the associated private key. Our focus will be on public-key cryptosystems.

Foundations in computational number theory

The difficulty in breaking any cryptosystem can be generalized to the difficulty of certain problems in number theory. Two problems specifically, integer factorization and the discrete logarithm problem, lie at the heart of most modern cryptosystems. Integer factorization describes the factorization of a composite integer into its integer divisors which, when multiplied together, equal that initial integer:

$$\text{Suppose } n \in \mathbb{N},\ n \text{ is composite,}$$

$$\text{Find } p_1^{\alpha_1} p_2^{\alpha_2} \dots p_m^{\alpha_m} = n \text{ where } \alpha, p \in \mathbb{N} \text{ and } p \text{ is prime}$$

As seen in the equation, the integer factorization problem bears striking resemblance to the fundamental theorem of arithmetic, which states that any positive integer equals certain powers of its prime factors. Currently, no computationally feasible algorithm has been discovered to compute the factorization of any integer, however, it has not been proven that no algorithm exists [**?**]. The discrete logarithm problem also has no known feasible solution, but has not been proven to lack one entirely. Although the problem may be represented over the multiplicative group of integers [7] (as in the integer factorization problem), we will first describe it abstractly:

$$\text{Suppose a group } G,\ |G| = n,\ P,\ Q \in G,\ \langle P \rangle = G$$

$$\text{Find } \alpha = log_P(Q)$$

The choice of a group G allows for an additional degree of complexity, as its only requirement is cyclicity. We will examine two common types of groups chosen for G: subgroups of the multiplicative integers, and subgroups of elliptic curves [2, p. 13].

Integer factorization

Integer factorization is used for several notable public-key cryptosystems, but no symmetric-key systems. The most notable example, RSA, will be explained below.

RSA

The RSA cryptosystem laid the foundation for public-key cryptography, and is still being used to this day. It utilizes several theorems and algorithms from classical number theory which form a seemingly-infeasible computational problem. The procedure for encryption and decryption is as follows:

Generate two large primes p and q

Let $n = p \cdot q$

Let $m = \phi(n) = (p-1)(q-1)$ where $\phi(x)$ denotes Euler's phi function

Generate a random number e such that $1 < e < n$ and $\gcd(e, m) = 1$

Using the Euclidean algorithm, find a d such that $\gcd(e, m) = 1 = d \cdot e + m \cdot r$

Note that $e \cdot d \equiv 1 \pmod{m}$. Thus, for some $\alpha \in \mathbb{N}$, for which we assume that $\gcd(\alpha, n) = 1$ (simply meaning that $\alpha \neq p, q, n$).

$$\begin{aligned}
\alpha^{e\cdot d} \pmod{n} &\equiv \alpha^{m\cdot r+1} \\
&= \alpha^{\phi(n)\cdot r+1} \\
&= (\alpha^{\phi(n)})^r \cdot \alpha^1 \\
&= (\alpha^0)^r \cdot \alpha^1 \\
&= 1^r \cdot \alpha^1 \\
&= \alpha \pmod{n}
\end{aligned}$$

Therefore, if α were to represent the data to be encrypted, the data would be secure when taken to a power of e or d. The only way to decrypt α^e or α^d would be to know d and e respectively. The RSA algorithm generates e and n as components of the public key with d and m as components of the private key. The one fault in the prior algorithm is the rare case where $\gcd(\alpha, n) \neq 1$. In this case, the result will still be the same due to [3, p. 109].

The RSA algorithm provides the advantage of combining message verification with encryption. This means that, with only the one encryption/decryption algorithm, a message can be verified to come from a certain entity and can be encrypted such that only a certain entity can decrypt it. Suppose the following two algorithms for encryption and decryption respectively:

$$\text{Encryption: } x \equiv \alpha^e \pmod{n}$$

$$\text{Decryption: } \alpha \equiv x^d \pmod{n}$$

One entity sending data to another can decrypt the data with their private key, then encrypt this result with the recipient's public key. Having sent this data to the recipient, the recipient can then decrypt the data with their private key, then encrypt the result with the sender's public key, resulting in the original data. Hence, the recipient has verified the identity of the sender, and the data was secured so that only the proper recipient could decrypt it.

Computational difficulty of integer factorization

Although cracking the RSA algorithm has not been proven to be equivalent to integer factorization, the most successful methods of solving RSA treat it as such. The fastest-known algorithms for solving integer factorization is the number field sieve. It has a subexponential runtime, for an input n with $log_2\ n = r$ bits, of approximately $O(\sqrt[3]{r^2}^{2.923\cdot\sqrt[3]{r}})$ bit operations [2, p. 18]. Such a runtime is feasible for reasonably-small input (of 768 bits or less) [4], however, proves infeasible for any larger input. In comparison to other cryptosystems, 768 bits is actually quite a large key size. We will soon see cryptosystems who become unfeasible to crack with only 128 bits of input.

Discrete Logarithm Problem **(over a group of multiplicative integers)**

The discrete logarithm problem presents a different type of computational challenge for cryptography. The simplicity of the problem shifts its computational complexity to the complexity of the group structure. If the structure of the group is easily predictable, then, for some generator P and random member of the group Q, it is easy to compute which power α of P yields Q, such that $P^\alpha = Q$. In this section, we will discuss cryptosystems based on the discrete logarithm problem as applied to a group of multiplicative integers modulo p (where p is some prime), and will discuss an even more complex group (the points on an elliptic curve) after.

The ElGamal cryptosystem

The ElGamal cryptosystem is a public-key cryptosystem based on the difficulty of the discrete log problem. Unlike RSA, ElGamal does not provide verification with its algorithm. However, ElGamal does in fact require less computational time for encryption and decryption than RSA, making it favorable for certain circumstances. Its algorithm is as follows:

1. Suppose a group $\mathbb{Z}_n$, $P, Q \in \mathbb{Z}_n$, $\langle P \rangle = \mathbb{Z}_n$, $Q = P^\alpha$, and all information but α is public

2. The sender chooses some random integer k, $1 < k < n-1$, and then send the tuple $(P^k, x \cdot Q^k = x \cdot P^{\alpha \cdot k})$

3. The recipient, who generated α computes $(P^k)^{n-1-\alpha} \equiv (P^k)^{-\alpha} = P^{-\alpha \cdot k} \pmod{n}$

4. Then, $x \cdot P^{\alpha \cdot k} \cdot (P^k)^{-\alpha} = x$

This cryptosystem enables a message x to be sent in such a way that it is only readable if the recipient knows α in the equation $Q = P^\alpha$, hence, ElGamal is assumed to be as secure as the discrete logarithm problem. ElGamal is primarily used in conjunction with a symmetric-key cryptosystem, where the ElGamal algorithm is used to share a secret key, and the symmetric system is used henceforth. This is an example of a hybrid cryptosystem. For another example, we turn to another early example of public-key cryptography, the Diffe-Hellman key exchange protocol.

Diffe-Hellman key exchange

The Diffe-Hellman protocol bears striking resemblance to ElGamal due to their basis in the discrete logarithm problem. Because it is solely used in conjunction with a symmetric-key cryptosystem, it specifically relies on the participation of two entities. Its algorithm is as follows:

1. Both entities agree on a group $\mathbb{Z}_p$ and a generator α

2. One chooses a large and random integer x, $1 < x < n$, and sends α^x to the other

3. The other chooses a large and random integer y, $1 < y < n$, and sends α^y to the first

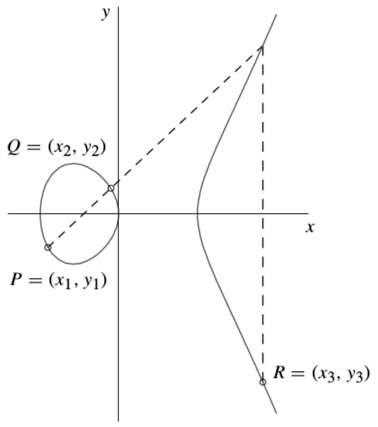
4. They take each other's public key and raise it to their own power: $(\alpha^x)^y = (\alpha^y)^x = \alpha^{x \cdot y}$

5. $\alpha^{x \cdot y}$ is now the shared secret key, only known to the two entities

Computing $\alpha^{x \cdot y}$ knowing only α^x, α^y, and α is assumed to be (but has not been proven to be) as hard as the discrete logarithm problem. To most-efficiently solve the discrete logarithm problem for a group of multiplicative integers, similar techniques to RSA are used. The number-field sieve is the fastest-known algorithm, and requires roughly $O(\sqrt[3]{r^2}^{2.923\cdot\sqrt[3]{r}})$ bit operations. Now, we will demonstrate the discrete logarithm problem with a different group to provide for improved computational security.

Discrete Logarithm Problem (over a group defined by an elliptic curve)

A group is referred to as an elliptic curve if the members of the group are points satisfying an equation of the form $y^2 = x^3 + ax + b$ and the operator of the group is the operator for adding points on a curve. This operator can be summarized as follows, where the first formula denotes the addition of two different points, and the second denotes the addition of a point with itself:

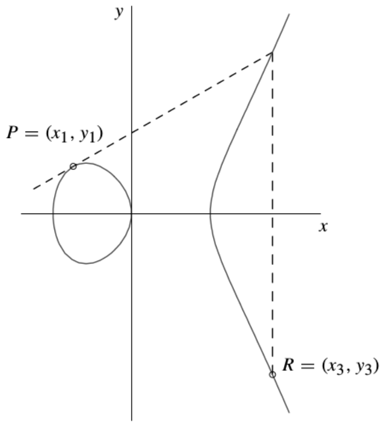


Addition: $P + Q = R$. [2]

$$P + Q = (x_1, y_1) + (x_2, y_2) = (x_3, y_3) = \quad R$$

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2$$
$$y_3 = -y_1 + \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3)$$

[6, p. 120]



Doubling: $P + P = R$. [2]

$$P + P = 2P = (x_1, y_1) + (x_1, y_1) = (x_3, y_3)$$

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1$$
$$y_3 = -y_1 + \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3)$$

[6, p. 120]

For a more in-depth discussion of elliptic curves, see [2, c. 3], [5, p. 167], or [6, p. 117].

The implications of substituting an elliptic curve into the discrete log problem are vast. The minimum security level required of RSA/ElGamal has a key size of 1024 bits, which is 6.4 times larger than an equally-secure key size (160 bits) for an elliptic curve cryptosystem [2, p. 19]. The gap between the security of RSA/ElGamal and elliptic curve systems grows as the number of bits increases because the most-efficient known algorithm for solving elliptic curve cryptosystems (Pollard's rho algorithm; for an explanation of the algorithm, see [2, p. 157]) is less efficient than the number field sieve (in most cases). Moreover, because the operator for an elliptic curve is so much more complected than multiplication of integers, encryption and decryption algorithms are less efficient for elliptic curve cryptosystems [2, p. 19].

Conclusion

In short, we conclude that the "best" cryptosystem depends on the longevity of the security of the data, the size of the data being transmitted, the computational complexity of encryption and decryption, and the desired convenience of the algorithm. For longest-term security, it is best to use an elliptic curve cryptosystem with a large (376 bits or more [1]) key size, however, the importance of using an elliptic curve system diminishes in proportion to the importance of storage size (e.g. in cases where two parties are sharing a secret key). For sending the smallest amount of data, once more, use an elliptic curve cryptosystem, for they are far superior in this regard. For computing encryption and decryption most efficiently, use a cryptosystem which only requires taking powers of integers, such as ElGamal or RSA. And finally, for greatest convenience, use a purely public-key algorithm so that all keys for communication are pre-established, hence, no secret key needs to be shared when two parties wish to communicate. By using a combination of these four factors, one can optimize their choice of a cryptosystem to best suit their needs and secure their data.

References

[1] Elaine Barker, Don Johnson, and Miles Smid. Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography. *NIST Special Publication*, pages 800–56A, 2007.

[2] Darrel Hankerson, Scott Vanstone, and Alfred J Menezes. *Guide to elliptic curve cryptography*. Springer, 2004.

[3] Thomas W Judson. *Abstract algebra: theory and applications*. Stephen F. Austin State University, 2014.

[4] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen Lenstra, Emmanuel Thom, Joppe Bos, Pierrick Gaudry, Alexander Kruppa, Peter Montgomery, Dag Arne Osvik, Herman te Riele, Andrey Timofeev, and Paul Zimmermann. Factorization of a 768-bit rsa modulus. 2010. `http://eprint.iacr.org/`.

[5] Neal Koblitz. *A course in number theory and cryptography*, volume 114. Springer, 1994.

[6] Neal Koblitz. *Algebraic aspects of cryptography*, volume 3 of *Algorithms and computation in mathematics*. Springer, 1998.

[7] Alfred J Menezes, Tatsuaki Okamoto, and Scott A Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *Information Theory, IEEE Transactions on*, 39(5):1639–1646, 1993.

[8] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 2010.