|  |
| --- |
| Hochschule Darmstadt, EIT |
| Cryptography Algorithms |
| Comparison of Public Key Algorithms and Best available support libraries |

|  |
| --- |
| Hari Krishna Yelchuri  5-4-2020 |

Contents

[1. Public Key Cryptography Algorithms 2](#_Toc39477009)

[1.1 RSA 2](#_Toc39477010)

[1.2 DSA 2](#_Toc39477011)

[1.3 ElGamal Encryption Algorithm 3](#_Toc39477012)

[1.4 Diffie-Hellman key exchange 3](#_Toc39477013)

[1.5 Elliptical Curve Cryptography (ECDH, ECDSA) 3](#_Toc39477014)

[1.6 EdDSA 3](#_Toc39477015)

[2 Comparison of Algorithms 3](#_Toc39477016)

[3 KEY CONCLUSIONS 4](#_Toc39477017)

[4 SUPPORT LIBRARIES 4](#_Toc39477018)

# Public Key Cryptography Algorithms

With the spread of more unsecure computer networks in last few decades, a genuine need was felt to use cryptography at larger scale. The symmetric key was found to be non-practical due to challenges it faced for key management. This gave rise to the public key cryptosystems.

The most important properties of public key encryption scheme are –

* Different keys are used for encryption and decryption. This is a property which set this scheme different than symmetric encryption scheme.
* Each receiver possesses a unique decryption key, generally referred to as his private key.
* Receiver needs to publish an encryption key, referred to as his public key.
* Some assurance of the authenticity of a public key is needed in this scheme to avoid spoofing by adversary as the receiver. Generally, this type of cryptosystem involves trusted third party which certifies that a particular public key belongs to a specific person or entity only.
* Encryption algorithm is complex enough to prohibit attacker from deducing the plaintext from the ciphertext and the encryption (public) key.
* Though private and public keys are related mathematically, it is not be feasible to calculate the private key from the public key. In fact, intelligent part of any public-key cryptosystem is in designing a relationship between two keys.

Let’s discuss a few widely used public key algorithms:

## RSA

**RSA** (**Rivest–Shamir–Adleman**) is one of the first public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and distinct from the decryption key which is kept secret (private). In RSA, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers, the "factoring problem".

A user of RSA creates and then publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret. Anyone can use the public key to encrypt a message, but only someone with knowledge of the prime numbers can decode the message. Breaking RSA encryption is known as the RSA problem. Whether it is as difficult as the factoring problem is an open question. There are no published methods to defeat the system if a large enough key is used.

## DSA

The DSA algorithm works in the framework of public-key cryptosystems and is based on the algebraic properties of modular exponentiation, together with the discrete logarithm problem, which is considered to be computationally intractable. The algorithm uses a **key pair** consisting of a **public key** and a **private key**. The private key is used to generate a **digital signature** for a message, and such a signature can be **verified** by using the signer's corresponding public key. The digital signature provides message authentication (the receiver can verify the origin of the message), integrity (the receiver can verify that the message has not been modified since it was signed) and non-repudiation (the sender cannot falsely claim that they have not signed the message).

## 1.3 ElGamal Encryption Algorithm

In cryptography, the **ElGamal encryption system** is an asymmetric key encryption algorithm for public-key cryptography which is based on the Diffie–Hellman key exchange. It was described by Taher Elgamal in 1985. The Digital Signature Algorithm (DSA) is a variant of the ElGamal signature scheme, which should not be confused with ElGamal encryption.

## Diffie-Hellman key exchange

**Diffie–Hellman key exchange** is a method of securely exchanging cryptographic keys over a public channel and was one of the first public-key protocols as conceived by Ralph Merkle and named after Whitfield Diffie and Martin Hellman. DH is one of the earliest practical examples of public key exchange implemented within the field of cryptography.

Traditionally, secure encrypted communication between two parties required that they first exchange keys by some secure physical means, such as paper key lists transported by a trusted courier. The Diffie–Hellman key exchange method allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.

## Elliptical Curve Cryptography (ECDH, ECDSA)

Elliptical curve cryptography (ECC) is a public key encryption technique based on *elliptic curve theory* that can be used to create faster, smaller, and more efficient cryptographic keys. ECC generates keys through the properties of the elliptic curve equation instead of the traditional method of generation as the product of very large prime numbers. The technology can be used in conjunction with most public key encryption methods, such as RSA, and Diffie-Hellman. According to some researchers, ECC can yield a level of security with a 164-bit key that other systems require a 1,024-bit key to achieve. Because ECC helps to establish equivalent security with lower computing power and battery resource usage, it is becoming widely used for mobile applications.

## EdDSA

In public-key cryptography, **Edwards-curve Digital Signature Algorithm** (**EdDSA**) is a digital signature scheme using a variant of Schnorr signature based on twisted Edwards curves.[[1]](https://en.wikipedia.org/wiki/EdDSA#cite_note-RFC8032-1) It is designed to be faster than existing digital signature schemes without sacrificing security.

# Comparison of Algorithms

These algorithms differ in some way or the other with each other. The differences are as follows:

* RSA is an asymmetric algorithm that can be used for encryption and signing. The related algorithm for creating a shared secret symmetric key (key exchange) is Diffie-Hellman.
* RSA bases its security on the difficulty of integer factorization, while Diffie-Hellman uses the difficulty of finding discrete logarithms.
* Using RSA directly is often impossible, it works on integers and the operations are quite slow.
* That creates a need for padding (making messages the right size) and makes it impractical to encrypt or sign large amounts of data directly.
* These issues (and some others) make RSA fragile: it's easy to screw up the implementation of the padding system or the actual encryption/signing to leak data.
* DSA uses a very similar process to RSA signing internally, but specifies the particular key generation and hashing steps needed to create signatures properly.
* More importantly it bases its security not over the difficulty of integer factorization but on the difficulty of the discrete logarithm problem.
* DSA is only for signatures, not for encryption or key exchange.
* Unfortunately, DSA and ECDSA are fragile: it requires a value to be chosen at random for each signature created. Failure to do this properly results in leaking the private key.
* ECDSA is very similar to DSA, but uses elliptic curve cryptography (ECC). ECC uses a different trapdoor function than RSA.
* Its security is based on the difficulty of the elliptic curve discrete logarithm problem.
* It allows for smaller key sizes than DSA, and is a good bit faster, but suffers from the same fragility as DSA (and RSA).
* EdDSA is an alternative to ECDSA designed not to have the fragility issues that DSA/ECDSA/RSA have. It's also faster. If you have a choice of signature algorithms this is the one to use.
* X25519 is a form of Elliptic-Curve Diffie-Hellman (ECDH) algorithm. It's used for key exchange. It's very fast and quite robust. If you have a choice of key exchange algorithms this is the one to use.

## KEY CONCLUSIONS

There are a few features that these algorithms are suitable for:

* RSA – Encryption, decryption and digital signature.
* ElGamal Encryption Algorithm – Encryption, Decryption (No signatures thus no authenticity)
* DSA – Digital Signatures (no encryption or decryption, only verification)
* EdDSA – Digital Signatures Algorithm (Key generated using Edwards curve)
* Diffie Hellman – Good Key Exchange Algorithm
* ECDH – Key Exchange Algorithm (Uses elliptical curves for key generation)

Due to the very slow speed of even the very fast asymmetric systems compared to current symmetric systems it's pretty rare to need to encrypt something directly with an asymmetric system. Instead it's common to use a key exchange algorithm and then a symmetric cryptosystem to send the actual message. For example, **use EdDSA to prove identity, Elliptic-Curve Diffie-Hellman (ECDH) (X25519) to exchange keys, and a suitable algorithm for the actual encrypted messages.**

## SUPPORT LIBRARIES

All the libraries listed here support these algorithms and are fully supported by PSoC 64.

* OpenSSL
* WolfSSL
* WolfCrypt
* MbedTLS

WolfCrypt and MbedCrypt (now merged with MbedTLS) are especially designed to use the crypto hardware.