yubico

A Question of Key Length

Does Size Really Matter When It Comes To Cryptography?

White Paper

December 7, 2015

By Alessio Di Mauro

Copyright

Trademarks

Disclaimer

Contact Information

Yubico Inc
420 Florence Street, Suite 200
Palo Alto, CA 94301
USA
yubi.co/contact

yubico

Contents

Introduction

One of the most interesting and useful aspects of the YubiKey NEO, YubiKey NEO-n, YubiKey 4, and YubiKey 4 Nano is that they can act as a smart card and come pre-loaded with a bunch of interesting applications, such as an implementation of OpenPGP Card.

Many users like this functionality, but some question the key lengths. It's an expected cryptographic question and is worth examining in some detail. I will walk you through it.

OpenPGP is a standard that allows users to encrypt, decrypt, sign, and authenticate data. It is an open, standardized variant of PGP, available as a FOSS implementation in the form GNU Privacy Guard (GPG) and is most notably used for email encryption and authentication. Independent of the actual implementation, OpenPGP (and PGP) supports both symmetric and asymmetric cryptography. Today we will focus on the latter.

Simplified Cryptography Primer

To better understand what follows, a few very basic concepts of cryptography are required. In asymmetric, or public key encryption, there are two main players: the encryption algorithm itself (such as RSA, ECC, or ElGamal) and a cryptographic key pair.

Each encryption algorithm is normally based on a computationally hard problem. The mathematical transformation constitutes the operation that the encryption scheme can perform, such as *encrypt/decrypt* or *sign/verify*, whereas the keys provide the additional data.

The two keys of a given key pair are strongly interconnected; this is a fundamental property of asymmetric cryptography. The keys must be used together to achieve different properties such as confidentiality, authenticity, and integrity.

Confidentiality is a guarantee that the message is received only by the intended recipients. Authenticity guarantees the identity of the author, and integrity confirms both confidentiality and authenticity by ensuring that a message has not been modified in transit.

(For a more detailed explanation on cryptography, see the Addendum at the end of this white paper.)

On to PGP

All this can be achieved if, and only if, the secret key of a user remains uncompromised. However, not all keys are created equal.

In computer security, the strength of a cryptographic key is defined by its length measured in number of bits, rather than being connected to the number and shape of its ridges and notches like in a physical key (say for your car). Provided that an encryption algorithm actually supports different key lengths, the general rule is that the longer the key, the better.

Comparing Asymmetric Encryption Algorithms

There are many asymmetric encryption algorithms, but let's focus on RSA, which is one of the most popular and is supported by the YubiKey NEO, YubiKey NEO-n, YubiKey 4, and YubiKey 4 Nano. What is a suitable key length to use with RSA and why not just use the longest key possible?

RSA was first introduced in the '70s, but since it is based on a mathematically hard problem we are still able to use it with some adaptations.

Historically, a common starting point for a key length has been 1024 bits. Despite the fact that attacks on this class of keys are very sophisticated and targeted to specific platforms, 1024-bit keys are generally considered not secure enough and their use is highly discouraged.

In 2012, the National Institute of Standards and Technology (NIST), a US agency that promotes technological advancements, published a document that contained the following table (Table 4; page 67).

Security Strength		2011- 2013	2014-2030	2031-beyond
80	Applying	Deprecated	Disallowed	
	Processing	Legacy use		
112	Applying	Acceptable	Acceptable	Disallowed
	Processing			Legacy use
128	Applying/Processing	Acceptable	Acceptable	Acceptable
192		Acceptable	Acceptable	Acceptable
256		Acceptable	Acceptable	Acceptable

The column “Security Strength,” or more colloquially “Bits of Security,” is an estimation of the amount of work required to defeat a cryptographic algorithm, and therefore the higher the value, the better.

The keywords “Applying” and “Processing” refer to encryption and decryption operations respectively.

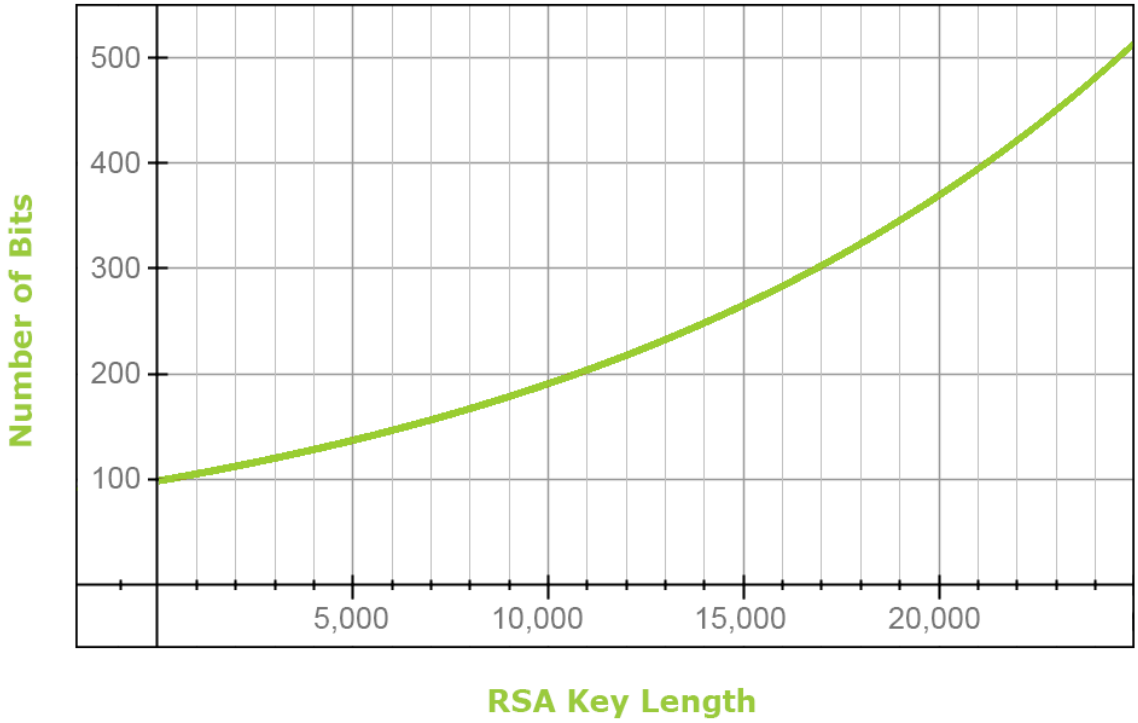
A Security Strength of 80 bits is currently *Disallowed* which translates to “an algorithm or key length [that] *shall not* be used for applying cryptographic protection.” Now, if you were guessing that 80 bits of security are approximately equivalent to RSA-1024, you have guessed right. This is mentioned in the same NIST document (Table 2, page 64).

Similar results also can be found in ECRYPT II, the second incarnation of ECRYPT, the European Network of Excellence in Cryptography (Tables 7.2 and 7.3 on page 30 of the NIST document). For clarity, in the following text we will use the data from the NIST publication.

The next relevant value in the table is 112 bits of security, which roughly corresponds to RSA with a key length of 2048 bits. At the moment this value is considered *Acceptable*, which means that it is not known to be insecure and it is deemed to be acceptable until 2030.

Now comes the interesting bit. Although there is no requirement to use RSA keys with a length that is a power of two, depending on the implementation there might be some advantages in terms of speed.

For this reason we take into account a length of 4096. Unfortunately, this value is not included in the table previously depicted. However, with a bit of exponential regression and assuming that the “Security Strength” function is continuous (or better, derivable) between the data points provided in the table above, we get the following plot:



As you can see, a 4096-bit RSA key clocks in at around 129 bits of security.

This value is marginally better than a key length of 3072 bits, and considered *Acceptable* beyond year 2030. (Also see this key-length calculator.)

The Big Debate: 2048 vs. 4096 and Yubico's Stand

Are 2048-bit keys useless? Are your documents completely insecure if you are using them? What are the pros and cons of one key length versus the other?

As I have shown, RSA-2048 still has fifteen years of life left before it is considered obsolete. There is plenty of time not to be worried now. Just imagine where technology was fifteen years ago!

While it is true that a longer key provides better security, we have shown that by doubling the length of the key from 2048 to 4096, the increase in bits of security is only 18, a mere 16%. Moreover, besides requiring more storage, longer keys also translate into increased CPU usage, higher power consumption, and slower operations.

While this might not seem much on a modern computer where we measure things in the order of gigabytes and hundreds of watts, it is still a valid concern for the ever-increasing market of low-power embedded devices, where CPU frequency is measured in kilohertz and power consumption in milliwatts and microwatts.

In these cases, using a longer key means longer time to compute the result and shorter battery life on devices.

The real advantage of using a 4096-bit key nowadays is future proofing, but by the time even RSA-2048 is declared dead, hopefully Elliptic Curve Cryptography (ECC) will have taken over or, even better, new and wonderful encryption algorithms will have been discovered.

What About ECC

So what about Elliptic Curve Cryptography? These encryption schemes are an alternative to RSA and are based on a completely different mathematical problem. Apart from that, however, they are just normal asymmetric encryption algorithms.

On the other hand, when it comes to speed and memory, ECC considerably outperforms RSA (with the notable exception of signature verification, where RSA is faster), even on embedded system and smaller microcontrollers.

Key lengths for these kinds of algorithms are considerably smaller. According to NIST, 112 and 128 bits of security, (equivalent to RSA-2048 and RSA-4096) correspond to 255-bit and 383-bit long ECC keys (worst case, even less on some specific curves).

So why are we not using this everywhere? Although the math behind them has been known for a while, ECC is a relatively new concept in cryptography, an inherently slow-changing and conservative field.

New implementations and new "fast reduction" curves that make computation significantly quicker are still under study and it takes time. As if that was not enough, some curves and implementations are behind patent walls.

Support for these algorithms in OpenPGP has been proposed, and the first implementations are slowly starting to appear. Implementing cryptography, however, is an error-prone procedure and a fine art in and of itself.

Blindly implementing an algorithm is usually not enough to plug all the potential security holes, and be impervious to side-channel attacks and the like.

It is clear that once the issues are resolved and more implementations start coming around, the debate may shift to ECC or quantum-resistant algorithms.

Where Does Yubico Stand?

The YubiKey NEO and YubiKey NEO-n implement OpenPGP and support RSA up to 2048 bits. The YubiKey 4 and YubiKey 4 Nano, introduced in November 2015, also implement OpenPGP but add support for RSA 4096-bit keys.

There is also the possibility of support for ECC, but it cannot be easily implemented without using some proprietary extensions. Moreover, as stated before, implementing crypto is a difficult process and although we have an initial version available on GitHub, this still requires more thorough testing before it is considered production-ready.

A best practice is to determine how long you plan to use a specific key and then select a key length based on that decision. Everyday smart cards are fine at 2048 bits because they get changed out at regular intervals and will naturally migrate to longer key lengths over time. Long-term keys, such as your master OpenPGP key that isn't on a smart card or used every day, could be viable for the next 30 years if you pick longer key lengths today.

All in all, we believe that the security of the asymmetric cryptography provided by the YubiKey NEO and YubiKey NEO-n is adequate for the time being. However, we are constantly working to keep ourselves ahead of the curve (no pun intended) and our new YubiKey 4 and YubiKey 4 Nano prove that, with their support for 4096-bit keys.

Addendum

A Crash Course in Cryptography

To better understand asymmetric cryptography, you need knowledge of some basic concepts.

For those who are not familiar with public key cryptography, we will provide here a *brief*, stripped-down introduction to the topic.

In asymmetric or public-key encryption there are two main players: the encryption algorithm itself (such as RSA, ECC, ElGamal) and a cryptographic key pair (there are also encryption/signature schemes such as PKCS#1, ECDSA and ECDH, but that is another discussion). The former is something that is (*or should be…*) publicly available. It tells us what are the steps to follow in order to encrypt and decrypt messages.

A public/private key pair on the other hand is part of the input to the encryption algorithm and provides two things: the information necessary to uniquely identify a user (public key), and a connected secret required to make the scheme secure (private key).

How does this all work together? Each encryption algorithm is normally based on a computationally hard problem. That is, some kind of mathematical operation that can be performed and inverted relatively easily provided that some information is available. The mathematical transformation constitutes the operation that the encryption scheme can perform, *encrypt/decrypt*, whereas the keys provide the additional data.

The two keys of a same key pair are strongly interconnected. If the public key is used as part of a message transformation, only the private key can be used to invert it and obtain the same data back. This is a fundamental property of asymmetric cryptography and, depending on how the transformations are applied, and as long as the private key remains so, it allows us to achieve different properties such as confidentiality, authenticity, and integrity.

Confidentiality is the guarantee a message will only be received (in a meaningful state) by its intended recipients. This is achieved by *encrypting* the message with the public key of the recipient, so that only she will be able to *decrypt* it with her private key.

Authenticity, on the other hand, guarantees the identity of the author and can be achieved by *signing* a message with the private key of the author and *verifying* it with his public key.

Finally, integrity is a somewhat orthogonal property, necessary for both confidentiality and authenticity to be upheld. It can guarantee that a message has reached a recipient (intended or not) unmodified. A typical way of providing integrity is through message authentication codes (MACs).