Security Engineering

Cryptography and Information Security
Master in Informatics

José Bacelar Almeida

Applied Cryptography

Symmetric Crypto

Asymmetric Crypto

Applications

Roadmap

- Key-Agreement
  - Diffie-Hellman protocol
- Public-key cryptography
  - Public-key Encryption (PKE)
  - Digital Signatures (PKS)
  - Asymmetric primitives
- Elliptic-Curve Cryptography (ECC)
- Cryptography and Quantum Computation
  - Post-Quantum Cryptography (PQC)

Key-Agreement

Context

- Symmetric crypto rely on shared secret-keys;
- Pre-agreement of keys is a costly procedure (it requires the use of secure channels…), and inflexible (e.g. consider adding an agent to the community…).
- Are there viable alternatives?
    - E.g.: Suppose we have a (symmetric) cipher in which the cipher operation is commutative, i.e. $E_{k1}(E_{k2}(X))=E_{k2}(E_{k1}(X))$
    - Each party (*A* and *B*) generate a secret key (*KA* and *KB*, respectively)
    - For *A* to communicate *M* with *B* it can
        - *A* sends $E_{KA}(M)$ to B — (note that *KA* is only known by *A*);
        - *B* returns $E_{KB}(E_{KA}(M))=E_{KA}(E_{KB}(M))$ to *A* — (again, *KB* is only known by *B*);
        - *A* decrypts the received message, and resends to *B* the result $E_{KB}(M)$;
        - *B* finally decrypts the ciphertext with its own key, obtaining message *M*.

    ... that is, *A* and *B* communicate securely without sharing secrets... (message *M* is always protected with at least one layer of encryption).
- Remark: but this scheme also displays important vulnerabilities... (c.f. man-in-the-middle attack studied below)

Key-Agreement

- The problem of key distribution can be circumvented if both parties agree on a common secret...
    - ...exchanging messages on a public channel...
    - ...but without it being possible to derive the secret knowing only the messages exchanged.
- A scheme that accommodates these requirements appeared in the article (New Directions in Cryptography, Diffie & Hellman 1976).
- Security follows from one-wayness of modular exponentiation (hardness of discrete logarithm).

Intermezzo: Algebraic Structures

- When p is a prime number, $GF(p)=(Z_p, +, *)$ is a **field** (+ and * are respectively addition and multiplication modulo p).
  - Modular inverses can be computed by the (extended) Euclidean algorithm;
  - A primitive element is a **generator** g of the multiplicative group (that is, each non-zero element can be written as g^i, for some i.
- For composite n, $(Z_n, +, *)$ forms a **ring**.
  - Elements with multiplicative inverse (**units**) are those x such that $gcd(x, n)=1$ (*primes relative* to n).
  - The number of those elements are given by $\varphi(n)$ (Euler function).

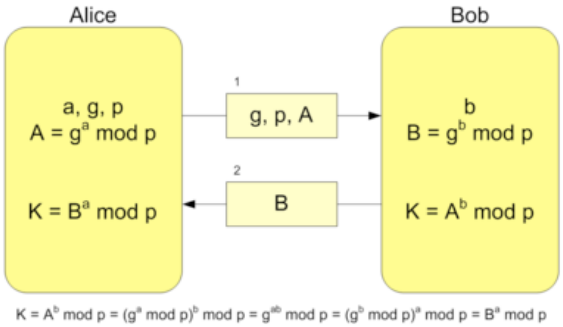
(Believed) Hard problems

Under appropriate conditions, the following problems are considered difficult.

- **Integer factorisation**
  - Given an integer n, determine its prime number factorisation. In other words, determine the prime numbers $p_1,...,p_i$ such that $p_1 \times ... \times p_i = n$.
- **Discrete logarithm**
  - Given a, b and n, compute x such that $a^x \bmod n = b$.
- **Discrete square-root**
  - Given y and n, compute x such that $x^2 \bmod n = y$.

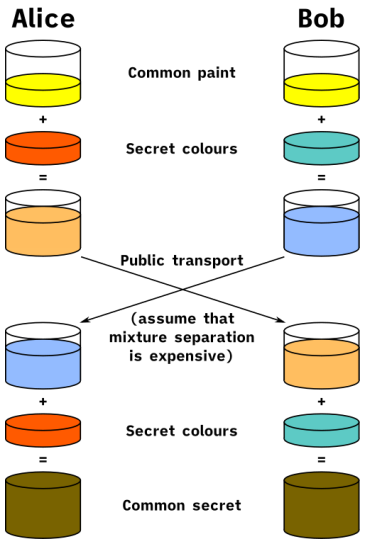
(Ephemeral) Diffie&Hellman protocol

- Parameters:
  - Let p be a prime and g a generator of a subgroup of Z_p^* of prime order q.
- Description:
  - A randomly chooses an integer $\boldsymbol{sk_A}=x$ s.t. $1<x<q$ (**private key**), sending $\boldsymbol{pk_A}=g^x \bmod p$ (**public key**) to B.
  - Similarly, B randomly chooses $\boldsymbol{sk_B}=y$ s.t. $1<y<q$, sending $\boldsymbol{pk_B}=g^y \bmod p$ to A.
  - **Shared secret**: $\boldsymbol{K}=g^{xy} \bmod p = (g^y)^x \bmod p = (g^x)^y \bmod p$.
- Notice that each party generates a key-pair for each execution of the protocol — hence the qualifier *Ephemeral*.
- Remark: shared secret K should never be used directly as a cryptographic key. It should instead be feeded into a KDF to derive needed secrets.



DH security wrt passive adversaries

- If the adversary is able to compute the discrete logarithm, then it could compute x from g^x, thus attacking the protocol.
- Strictly speaking, the security of the protocol is expressed as a security assumption of its own — the *Decisional Diffie-Hellman* (DDH) problem: for random x, y and z, it is infeasible a PPT adversary to distinguish between g^{xy} and g^z.



DH (in)security wrt active adversaries

- An *active adversary* might impersonate another party and so compromise the secrecy of the shared secret: known as **man-in-the-middle attack**.
- Example:

  Suppose *A* wants to agree on a secret with *B:*

  - *A* generates a key-pair $(sk_A, pk_A)=(x, g^x)$, and sends pk_A to *B*;
  - *I* intercepts *A*'s message;
  - *I* generates its own key-pair $(sk_I, pk_I)=(z, g^z)$, returning pk_I to *A*:
  - *A* adopts the secret $K=(g^z)^x=g^{xz}$ which it presumes agreed with *B*;
  - *I* knows the secret $K=(g^x)^z=g^{xz}$ that *A* believes is shared with *B*.
- Most asymmetric cryptographic techniques are vulnerable to this attack!

the use of asymmetric cryptography depends on a reliable association between public-key and legitimate parties (identities).

Public-Key Cryptography

Public-Key Cryptosystems

- The article that introduced the DH protocol suggested other possible applications of the underlying mechanism: the splitting the key into two components (a **key pair**):
    - one that remains secret — **secret-key** (**sk**)
    - another that can be made public — **public-key** (**pk**), since knowing it does not allow the corresponding secret key to be computed.
- Notice that:
    - secret keys never need to (and cannot) be shared
    - We don't have to guarantee the secrecy of the public keys. They can be transmitted over an open channel.
    - But questions about the authenticity of public keys remain, as was the case with the DH protocol…

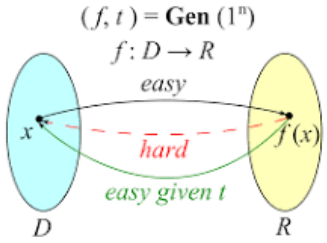
Public-Key Encryption

- We can extrapolate the impact of the changes mentioned above on techniques such as encryption — **Public-Key Encryption (PKE)**.
    - *Bob* generates a key-pair: $(sk_B,pk_B) = KeyGen()$
    - *Bob* sends pk_B *to Alice* (or Alice accesses it by some mean…)*;*
    - *Alice* encrypts a message to *Bob* using his public-key: $C = Enc((pk_B,M)$
    - *Bob* decrypts it using his private-key: $M' = Dec(sk_B,C)$
- Clearly, we expect that:

$$Dec(sk_B,Enc(pk_B,M)) = M \qquad \text{(correctness criterion)}$$

One-way trapdoors

- From a security perspective, it translates in a refinement of one-way functions with one additional requirement — a **trapdoor one-way permutation**:
  - encrypting with the public-key should be an one-way function
  - unless you know the corresponding private key (which allows you to do the reverse operation).



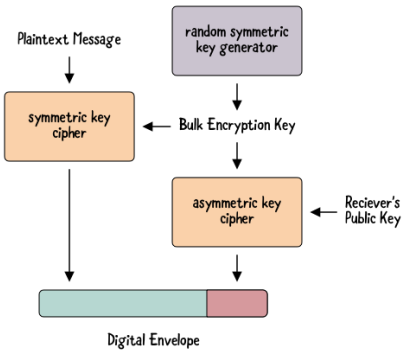
Man-in-the-middle (II)

- As mentioned, PKE does not escape the vulnerability non-authentic public-keys.
- It is vulnerable a man-in-the-middle attack:
  - *Intruder* intercepts *Bob*'s public-key (pk_B) and replaces by his own (pk_I);
  - *Alice* computes $C = Enc((pk_I,M)$, believing it can only be decrypted by *Bob*;
  - *Intruder* decrypts it using his private-key: $M' = Dec(sk_I,C)$;
  - ... and potentially send it again to *Bob.*
- It reinforces the mantra: asymmetric cryptography relies on authentic public-keys.

Hybrid Encryption

- A PKE scheme is only able to handle messages of a restricted shape.
- But unlike block ciphers, where modes of operation lift encryption to arbitrary messages, efficiency and security considerations lead to an alternative strategy.

  Hybrid Encryption combines the use of a symmetric cipher (that will encrypt the actual message), with an asymmetric PKE scheme that encrypts a randomly generated symmetric key.
- In its simpler form, it takes the structure of what is known as a **digital envelope.**



KEM/DEM paradigm

- The KEM/DEM paradigm structures the hybrid encryption strategy, paving the way for a comprehensive security analysis.
- A **KEM**/**DEM** consists of two components:
  - **Key Encapsulation Mechanism** (**KEM**) — a public-key scheme that is tailored to generate (symmetric) secret keys and their encapsulation by a PKE.
    - KeyGen : SecParam -> (SecKey, PubKey)
    - Encap: PubKey -> (SymKey, CipherText)
    - Decap: (SecKey, Ciphertext) -> SymKey
  - **Data Encapsulation Mechanism** (**DEM**) — a symmetric scheme that uses the KEM-generated key (e.g. an authenticated encryption scheme).

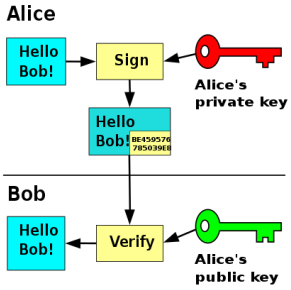
Digital Signatures

- Swapping the roles of public/secret keys in PKE make it possible to define a digital analogue of *document signing*.
- Often recognised as the main contribution of asymmetric cryptography.
- Properties:
  - *Integrity*: the message is not modified after signing;
  - *Authenticity*: the identity of the signer can be confirmed;
  - ***Non-repudiation***: it is possible to prove the identity of the signer.
- Recall that MACs were able to establish the first two properties…
- Note 1: the essence of the digital signature concept is an asymmetry between the capabilities of the verifier and the signer: the former must be able to verify the signatures produced by the latter without having the ability to produce them himself.
- Note 2: note that MACs guarantee the first two requirements but fail on the last (non-repudiation) - in this case the verifier has as much information as the signer. It is the ability of digital signatures to "authenticate documents" that make it possible to overcome the limitations of asymmetric techniques that have been identified...

Digital Signature Scheme

- Two entities: the *(**S**)igner* and the *(**V**)erifier*.
- A signature scheme comprises three operations:
  - keyGen*: secParam → secretKey * pubKey*
  - sign*: secretKey * message → signature*

    obs.: the signature is attached to the message — the *signed message* is thus the pair *(M,sig)*;
  - verify*: pubKey * (message * signature)* → Bool

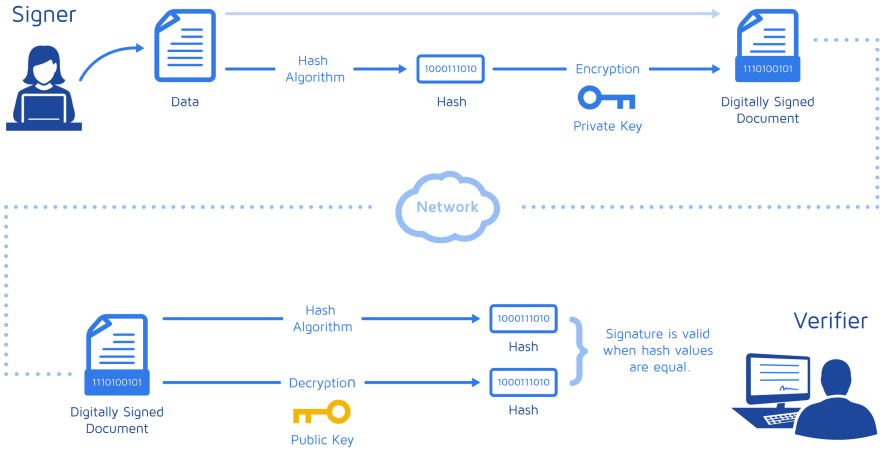


- Properties (for a signer's key-pair (sk^S, pk^S)):
  - **Correctness**: $\forall M$, verify(pk^S, (M, sign(sk^S, M)=`true`
  - **Security**: without knowing sk^S its impossible to forge a signature σ for any message M such that verify(pk^S, (M, σ))=`true`

Signatures and Hashing

- As with PKE, signatures are always used combined to symmetric primitives.

  Cryptographic Hash functions are used to map the content of arbitrary messages into digests (hash values)



Man-in-the-middle (III)

- But, as other asymmetric techniques, digital signatures are vulnerable to *man-in-the-middle* attacks.
- For signatures, this attack translates into the failure to authenticate the signer's identity upon verification (the verification is carried out with the "wrong" public-key).
- Note however that we are again at a point where we observe a circularity between what is the goal of the technique and the cause of the problem:
  - the digital signature is intended to establish the authenticity of a message/document;
  - and the failure to guarantee the authenticity of the association between public-keys and identities leads to the possibility of a man-in-the-middle attack.
- Hence:
  - Some trusted entity establish the association of identities and public-keys in some kind of "document"…
  - …and digital signatures would allow us to rely on the authenticity of those documents.

Public-Key Certification

- We have seen that asymmetric cryptography relies on the correct association between public-keys and agent identities. Of course, such a guarantee can be achieved by pre-distributing public-keys (but then we would not be far from the general key distribution problem…)
- More interestingly, we can use the very mechanisms provided by asymmetric cryptography (specifically, digital signatures) to establish public-key/identity associations:
  - All agents have the public-key of a trusted agent - the **Certification Authority** (**CA**). This public key must be obtained by some secure channel.
    - The CA guarantees (by digitally signing) the association between the agent's public key/identity - what we call a **public-key certificate**. The CA is responsible for ensure the correctness of the association.
    - Any entity can verify the signature of a certificate (thus attesting to the validity of the intended association).

Certificate usage

- The use of certificates adds a significant management burden to the use of public keys.
  - Certificate creation (e.g. for Alice):
    - Alice generates a key-pair;
    - Alice requests a certificate to a CA of his choice;
    - CA carries out due diligence to ensure the correctness of the information to be included in the certificate;
    - CA signs the certificate and sends it back to the user.
  - Certificate Use:
    - when Alice needs to give Bob her public key, she sends him her certificate;
    - Bob validates Alice's certificate and consults the public key it contains.
- Such requirements are only feasible for **long-term** key pairs.
  - This is indeed the case for key pairs used in PKE and Digital Signatures;
  - But not for (Ephemeral) Diffie-Hellman protocol.

Station-to-Station protocol

- The (Ephemeral) Diffie-Hellman protocol requires the key-pairs to be generated for each protocol execution.
- Instead of directly certifying the EDH public keys, one can instead rely on normal digital signatures (and corresponding certificates) to ensure the authenticity of each party.
- The resulting protocol is called **Station-to-Station:** protocol
  - A→B: $+x \,.\, g^x$
  - B→A: $+y \,.\, g^y$, $E_K(\text{sign}(sk^B, (g^y \| g^x)))$, $Cert^B$
  - A→B: $E_K(sign(sk^A, (g^x, g^y)))$, $Cert^A$
- Notation: "$A \rightarrow B: +x \,.\, M$" denotes a step in which A generates the value x and sends B the message M; $E_K(M)$ denotes the cryptogram resulting from encrypting M with the agreed shared key $K=g^{x*y}$; and $\text{sig}(sk^A, M)$ the digital signature of M by A (assumed to be properly verified upon reception).

Asymmetric Primitives

RSA primitive

- Algorithm that realises the concept of public-key cryptography introduced by *Diffie & Hellman*, proposed by *Ron **R**ivest, Adi **S**hamir & Leonard **A**dleman* (1977/8).
- Security is based on the difficulty of factoring integers.
- Description:
  - KeyGen():
    - generate two random primes p and q;
    - let $n=p*q$ (hence, phi(n)=(p-1)*(q-1), and choose e such that gcd(e,n)=1;
    - compute d such that $e*d=1 \bmod \text{phi}(n)$;
    - return (sk, pk), where the secret-key sk = (n, d); public-key pk = (n, e).
  - RSA($pk=(n, e)$, m) = $m^e \bmod n$
  - RSA^{-1}($sk=(n, d)$, c) = $c^d \bmod n$
- It can be used both for Public-Key Encryption and Digital Signatures.

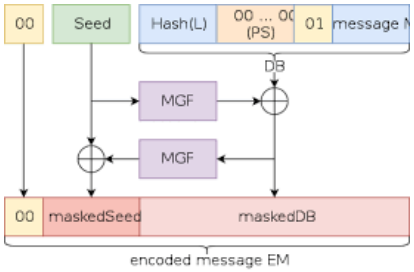
RSA security

1. Deriving the secret-key from the corresponding public-key was shown to be equivalent to factoring $n=p*q$.
2. Inverting RSA(pk, m) is believed to be difficult without knowing sk (RSA assumption) for a random m (in the range $0 \leq m < n$).
3. <u>However, is trivial to break if m is chosen from a small message space</u>!
   - This is in fact, a consequence of the determinism of RSA primitive (i.e. a given message encrypted repeatedly always results in the same cryptogram).

- Key sizes: a modulus of 3072 bit is believed to provide the security of AES-128.
  - From this we can extrapolate a big gap on the efficiency between symmetric and asymmetric cryptography — the former is typically thousands of times faster than the later (for a given security level).
- The determinism of RSA (as well as properties arising from the internal structure of the algorithm) means that <u>an appropriate padding scheme should always be used</u>.

RSA random variants

- The (asymmetric) padding used in RSA has two aims:
  1. add randomness to the encoded message;
  2. add redundancy to the message (built-in sanity check).
- Two schemes are widely used:
  - RSA-**OAEP** — used for Public-Key Encryption;
  - RSA-**PSS** — used for Digital Signatures.
- E.g. OAEP (Optimal Asymmetric Encryption Padding):



- Both (padded) schemes exhibit very strong security results (under the RSA assumption).

El-Gamal

- PKE algorithm introduced in 1984 by *T. El Gamal*.
- Security is based on the discrete logarithm problem.
- Description:

  Algorithm Parameters: a prime p and a generator g of prime order q.
  - KeyGen(p,g):
    - generate a random x s.t. $0 < x < q$;
    - lreturn (*sk*, *pk*), where the secret-key $sk = x$; public-key $pk = g^x$.
  - Enc(*pk, m*) = +*k*, (g^k mod *p*, ($m * pk^k$) mod *p*)
  - Dec(*sk*, *c=(a,b)*) = b / a^{sk} mod *p*
- Observations:
  - encryption is a randomised algorithm;
  - ciphertext doubles the size of the plaintext;
  - Key sizes: similar to RSA.

El-Gamal security

- Deriving the private key from the public key corresponds exactly to the discrete logarithm problem.
- Encrytion can be shown to be “semantically secure” under chosen plaintext attacks: an (PPT) adversary cannot distinguish the ciphertexts of two adversary chosen plaintexts (aka Ciphertext Indistinguishability under Chosen Plaintext — IND-CPA).

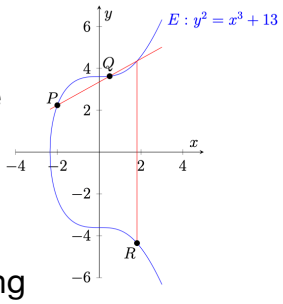
- A scheme with great academic relevance but little practical impact.

Digital Signature Algorithm (DSA)

- Signature algorithm included in the Digital Signature Standard (DSS) - 1991.
- Developed by the NSA for the NIST.
- Security based on the hardness of the discrete logarithm (like El-Gamal).
- Designed to provide a very efficient signature procedure — e.g. much more efficient than RSA... (on the other hand, verification is much heavier).
    - It is therefore particularly suited to running in environments with limited resources (e.g. smartcards).

Elliptic-Curve Cryptography

Elliptic-Curve Cryptography (ECC)



- The "discrete logarithm" problem can be expressed on any finite body (e.g. GF(p) or GF(p^n)).
- But it can also be "transposed" to a cyclic group defined on an elliptic curve defined on such fields.
- In Elliptic Curve Cryptography (ECC), the operation corresponding to modular exponentiation is “scalar-multiplication (k ⋅ P)”…
- …whose inverse operation is the “Elliptic Curve Discrete Logarithm (ECDL)”.
- ...which is also considered a difficult problem (at least if the underlying curve parameters meet certain criteria).
- Variants of the primitives based on the Discrete Logarithm problem can therefore be defined to operate on Elliptic Curves (e.g. ECDH; ECDSA; etc.).

ECC security

NIST guidelines for public key sizes for AES

ECC KEY SIZE (Bits)	RSA KEY SIZE (Bits)	KEY SIZE RATIO	AES KEY SIZE (Bits)
163	1024	1 : 6	
256	3072	1 : 12	128
384	7680	1 : 20	192
512	15 360	1 : 30	256

Supplied by NIST to ANSI X9F1

- The great advantage of elliptic curve cryptography is that it allows compact representations of the keys to achieve the desired security levels.
- A critical ((and controversial)) aspect of ECC is the choice of parameters (curve parameters and choice of base point).
    - The nature of the criteria required in these decisions, and the computational effort typically involved, means that the only viable way to take advantage of ECC is to use tabulated (standardised) parameters.
    - Possible choices:
        - NIST-P (e.g. P-256; P384; P-521) — (proposed by NSA);
        - Brainpool — “random” parameters proposed by BSI;
        - Curve 25519 — parameters proposed by D. Bernstein.

Cryptography & Quantum Computation

Quantum Computation & Cryptography

- Quantum computing relies on the laws and phenomena of quantum mechanics to solve computational problems..
- The consequence is that, in this new paradigm, certain problems that are believed difficult can be solved efficiently (or with significant efficiency gains).
- The impact on symmetric cryptography is significant, but not catastrophic. It stems from the quadratic speedup that results from Grover's algorithm for unstructured search.
- But in asymmetric cryptography the consequences are critical, since most of the "hard problems" adopted today now have efficient solutions (using Shor's algorithm), namely:
  - Factorisation (RSA)
  - Discrete logarithm (Diffie-Helmman, ElGamal, DSA)
  - Elliptic Curve Discrete logarithm(ECDH, ECDSA, EdDSA, etc.)

Predictions for the announced fate...

- Today's quantum computers are still a long way from having a real impact on the security of currently used algorithms/key-sizes.
- But the investment in quantum computing research is massive…
- … and is expected that by 2030 it will probably be possible to break RSA-2048.
- …which poses **a real threat today**:

  Sensitive encrypted information can be collected today so that it can be cracked with QC in the near future.
- These predictions have led standards organisations to promote the upgrade of cryptography to make it resistant to the threat of quantum computing.

Post-Quantum Cryptography (PQC)

- Overcoming the problems that QC poses for the security of current cryptographic techniques does not mean basing cryptography on quantum computers!
- **Post-Quantum** (or **Quantum-Safe**) **cryptography** refers to classical cryptographic techniques that are secure against quantum adversaries.
- The challenge is to find "difficult problems" that resist quantum capable adversaries.
- Domains where some problems are believed to be “Quantum Safe”:
    - Lattices
    - Multi-variable systems of equations
    - Hash-based Cryptography
    - Code-based Cryptography
    - Isogenies in Super-Singular Elliptic Curves
- …with different compromises on confidence on its security; signature and PK sizes; efficiency; etc.

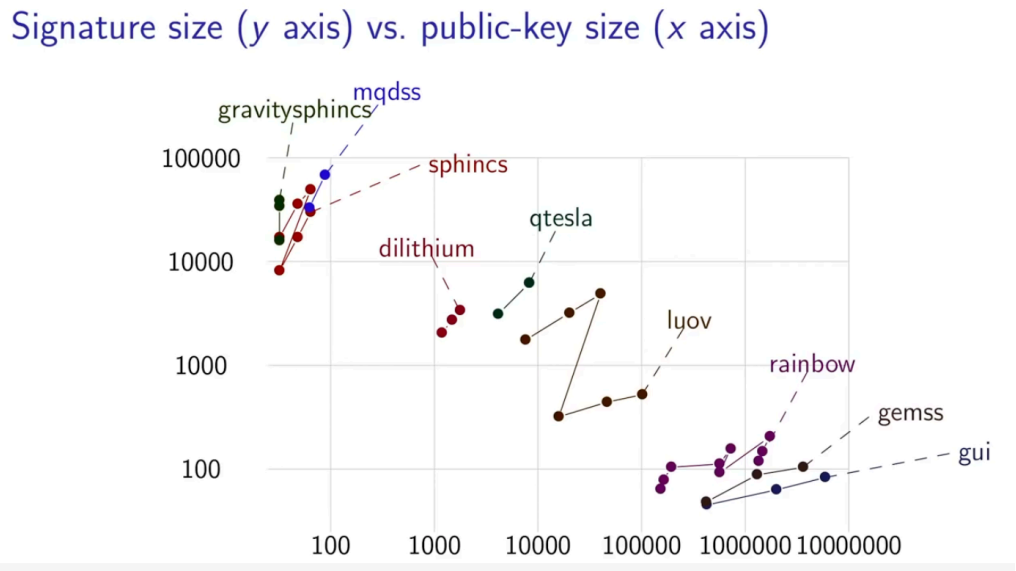
Transition to Quantum-Safe

- There has been a hasty effort to promote the transition to Quantum-Safe cryptographic techniques.
- But PQC needs "time" to become a viable alternative in cryptographic applications:
    - Trust (e.g. research in crypto-analysis)
    - New implementations
    - Usability and interoperability (e.g. redesign protocols, etc.)
    - Standards development
- On the other hand, transition should be as early as possible so that when a quantum computer is able to compromise some data, it is no longer sensitive.

PQC Standardisation process

- NIST initiative for PQC standardisation (announcement: 2016; final candidate selection: 2022)
- The nature of the NIST PQC call is different from previous calls (AES, SHA-3, …):
    - Post-quantum cryptography is more complicated!
    - Requirements/Timeline may vary depending on advances in the field.
    - There is not enough research in Quantum Computing to provide confidence regarding the different proposals...
    - "No silver bullet! — not expected to result in "pick a winner"
    - Ideally, a selection of "good alternatives"
    - Easy drop-in replacements are unlikely
- Standards:
    - Draft FIPS 203, *Module-Lattice-Based Key-Encapsulation Mechanism Standard (ML-KEM)* — [CRYSTALS-KYBER]
    - Draft FIPS 204, *Module-Lattice-Based Digital Signature Standard (ML-DSA)* — [CRYSTALS-Dilithium]
    - Draft FIPS 205, *Stateless Hash-Based Digital Signature Standard (SLH-DSA)* — [SPHINCS$^+$]
- Other standardisation initiatives are underway…

Sizes…



Hybrid Schemes

- The use of hybrid schemes is recommended at an early stage in the adoption of PQC.
- A Hybrid Scheme combines a classic cryptography with PQC:
  - E.g. for signatures: RSA-4096 + Dilithium-3
- Advantages:
  - Inter-operability/compatibility with "legacy" systems
  - Protects the risk associated with the novelty of the PQC technique
  - Security = at least as secure as the best of the combined schemes
- Disadvantages:
  - Still makes problems resulting from size more pronounced (public key, signature, etc.)
  - Redundancy ("computational overhead", etc.)

…from the perspective of Information Security

- (source: DHS/NIST post-quantum cryptography guidance):

1. Organizations should direct their Chief Information Officers to increase their engagement with standards developing organizations for latest developments relating to necessary algorithm and dependent protocol changes.
2. Organizations should inventory the most sensitive and critical datasets that must be secured for an extended amount of time. This information will inform future analysis by identifying what data may be at risk now and decrypted once a cryptographically relevant quantum computer is available.
3. Organizations should conduct an inventory of all the systems using cryptographic technologies for any function to facilitate a smooth transition in the future.
4. Cybersecurity officials within organizations should identify acquisition, cybersecurity, and data security standards that will require updating to reflect post-quantum requirements.
5. From the inventory, organizations should identify where and for what purpose public key cryptography is being used and mark those systems as quantum vulnerable.
6. Prioritizing one system over another for cryptographic transition is highly dependent on organization functions, goals, and needs.
7. Using the inventory and prioritization information, organizations should develop a plan for systems transitions upon publication of the new post-quantum cryptographic standard. Cybersecurity officials should provide guidance for creating transition plans.