Delphi: sharing assessments of cryptographic assumptions

Jeroen van de Graaf[1]* and Arjen K. Lenstra[2]

[1] Universidade Federal de Minas Gerais *and* ZKM
[2] none

Abstract. Almost all practical cryptographic protocols are based on computational or ad-hoc assumptions. Assessing the strengths of these assumptions is therefore a key factor in evaluating the risks of the systems using them. As a service to (and by) cryptographic researchers and practitioners, we propose to create *Delphi*, a public database where researchers document their opinions and beliefs about the strengths of the most important assumptions. We believe this effort will be of great value when deciding which cryptographic primitives to keep or start using.

1 Motivation

Cryptography is, well considered, nothing but a large house of cards: the overwhelming majority of algorithms and protocols are based on some underlying assumption which, if broken, will completely undermine the system's security. The only exceptions are primitives that come with a mathematical proof of security, such as those based on information theory (like the one-time pad) or quantum mechanics (quantum information theory, if you prefer), but none of these are generally considered to be sufficiently practical. Therefore, keeping track of the set of reliable assumptions and suitable parameters is a major responsibility for any professional crypto-practitioner.

Until recently a limited number of cryptosystems was widely used. Because changes in their underlying security assumptions were mostly gradual and predictable, a consensus emerged how cryptosystems could responsibly be chosen and parametrized [LV2001]. These days that is no longer the case. This is not so much due to the sheer number of assumptions to be considered, but because of the conceivable development of quantum computing, which is shaking up the cryptographic landscape. Even though it is unclear when or if this will ever happen, our security assumptions may have to be reassessed dramatically—or not at all. Expert opinions will cover a similarly broad range: the approach of [LV2001] that offered a single, generally supported view on each security assumption is certainly no longer representative and it is illusory to expect that agreement can be reached.

In this paper we propose to offer an alternative that should still, despite the lack of consensus, allow selecting or upgrading cryptosystems in an informed and responsible manner. We intend to do so by polling many non-anonymous experts on their opinions on a reasonably wide range of security assumptions. We will make the results available on our *Delphi* website, in such a way that, for each assumption, patterns (such as a *communis opinio* or apparent lack thereof) can easily be recognized.

In this paper we will focus on the most prominent, widely used assumptions, as specified in sections 3 and 4, after presenting some background in Section 2. Subsequently we solicit feedback (from as many experts as are willing to participate) on the questions formulated in Section 5. A finer-grained set of questions may later be considered targeting a wider range of assumptions and including specific cryptographic protocols if the idea of our oracle website catches on.

2 Background

2.1 Security levels – definition

Roughly speaking, the security level of a cryptosystem is an indication for the *complexity* of an *attack*, i.e., for the expected effort required to successfully undermine the cryptosystem's security.

* contact author: `jeroen.g@zkm.io`

The complexity of an attack against cryptosystem C is expressed by its **work factor**, i.e. the expected number $E(C)$ of *steps* to be carried out on a traditional computer or, depending on the context, on a quantum computer (where a step depends on C and on the type of computer; details are not relevant for our purposes). However, we often prefer to express security levels in terms of the logarithm base 2 of the work factor.

Definition 1 *The* security level $S(C)$ *of a cryptosystem* C *is the integer* k *for which* 2^k *is closest to* $E(C)$*.*

For any currently widely used cryptosystem C, the security level $S(C)$ cannot be guaranteed to be fixed over time, because it is always possible that a more efficient attack may be found. But even if $S(C)$ does not change, the effectiveness of the protection offered by C erodes over time because carrying out any fixed computational effort keeps getting easier and cheaper. Adequate values for $S(C)$ should resist this development (cf. below).

This leaves the issue how $S(C)$ is determined. An upper bound for $S(C)$ follows from the (proven or heuristically supported) complexity of an existing attack. To argue that C is secure, however, a lower bound for $S(C)$ is required. Deriving lower bounds is notoriously difficult, a problem that is conveniently dealt with by replacing it (for each widely used C) by one or two appropriate assumptions. These *security assumptions* are discussed further below. The practical upshot of this shortcut is that the best (i.e., lowest) upper bound known for $S(C)$ is used as lower bound as well, a rather unconventional approach that is inspired by ignorance and wishful thinking.

The above may be puzzling to an outside observer given the abundance of published “provably secure” cryptosystems, but strictly speaking, this terminology is misleading. Barring exceptions that the authors are not aware of, this all concerns in fact “provable security reductions” to a cryptosystem that is generally trusted to be secure. And this approach also has its problems; see [KM].

2.2 Security levels – adequate values

Adequate security levels evolve over time.

- In 1977 the trivial upper bound of 56 for the security level S(DES) of the *Data Encryption Standard* was interpreted as a lower bound too (cf. above) and deemed to provide adequate protection for commercial purposes by those who published DES. Since then S(DES) has not budged and is still believed to be 56, but since the 1990s breaking DES is considered to be easy.
- In 1992, a security level of 64 was still considered to be adequate (witness the publication of the cryptographic hash function MD5); nowadays security level 64 no longer suffices and, independently, MD5 *has* budged or, rather, has caved in (cf. section 2.4).
- In 1996 specialists estimated, in [BDRSSTW1996], that 90-bit security would suffice for the next 20 years. Even now an effort of 2^{90} operations is still far out of reach for academic approaches (and would require a colossal energy expenditure), with the effort of one of the largest public computational projects, the *Great internet Mersenne prime search*, hovering around 2^{77} operations per year.
- These days governments, state actors, and large corporations can make much larger efforts than academic cryptanalysts. Nevertheless, it is easily argued that on traditional computers breaking 128-bit security is quite a stretch for (or, rather, entirely out of reach of) even the most powerful adversary and thus adequate for the foreseeable future: with 2^{37} attackers (an optimistic estimate given a world population of 2^{33} people) each performing $2^{59} = 2^{25} * 2^{34}$ operations per year ($\approx 2^{25}$ seconds) on a traditional computer that carries out $16G = 2^{34}$ operations per second, on the order of $2^{128-37-59} = 2^{32}$ years, would be required for a single attack.

Brian Snow [BS], NSA's representative during the Advanced Encryption Standard (AES) selection process, agreed [BSPC] that 128-bit security is adequate for the long term future. Thus, he mentioned, a 256-bit key block cipher had to be standardized along with the new 128-bit one, to be ready to deal with the potential realization of quantum computing—then believed to be as imminent as now. According to Snow "the intermediate 192-bit security level is not good for anything because it is too much for traditional and too marginal for quantum computing; we standardized it only because it is right between 128 and 256".

2.3 Security assumptions

Two types of security assumptions are distinguished: computational and ad hoc security assumptions. All computing in the remainder of this section refers to a traditional computer.

Computational assumptions A computational assumption is the assumption that a particular computational problem cannot be solved efficiently. Many such problems are known, but so far there are just a handful that are useful for cryptography. This includes several "classical" problems that have been studied extensively for centuries, long before anyone realized their cryptographic potential. Although this may inspire confidence in their hardness, lacking lower bounds it should be kept in mind that this confidence is solely based on the fact that so far no-one has been able to find a sufficiently fast solution.

Computational assumptions mostly apply to public key cryptosystems, and the same computational assumption can be used for several distinct cryptosystems.

A cryptosystem that relies on a computational assumption can be broken by solving the assumed-to-be hard problem. The converse may not hold, i.e., the cryptosystem may be breakable without solving the hard problem. Indeed, for any computational assumption that underlies the security of a cryptosystem in wide use, usually an additional assumption is made, namely that the computational assumption suffices for the security of the cryptosystem as well. As an example, even though the ability to factor integers implies that RSA can be broken, it is not known if breaking RSA implies an efficient algorithm for factoring, strictly speaking. For now we have decided not to include questions related to this discrepancy.

Ad hoc assumptions An ad hoc assumption is the assumption that one particular cryptosystem has the security level that it was designed to have. Ad hoc assumptions mostly apply to block and stream ciphers and cryptographic hash functions, with each cipher or hash function relying on its own specific ad hoc assumption. Unravelling a problem created especially for a specific cryptosystem tends to be a problem of a less general and structured mathematical nature than any of the earlier computational problems. Although this by no means implies that it is easier, it less clear what precisely would inspire confidence in the hardness of the problem (or, equivalently, the security of the cryptosystem) other than the reputation and experience of its creator(s) and its peer reviewers, and to what extent "standard" or "believed to be effective" design approaches were followed.

2.4 Changes in the security assumptions

The unavoidable gradual erosion of security levels (due to faster and cheaper computing) was discussed above. Independently, new algorithmic, mathematical or cryptanalytic insights may affect security assumptions. So far this happened mostly gradually (cf. estimates in [LV2001]) and rarely caused cryptographic emergencies by entirely wiping out a cryptosystem: it happened, for instance, in 1984 to the Merkle-Hellman knapsack public key system, in 2004 to the very widely deployed MD5 cryptographic hash function (taking 4 years to sink in), and in 2022 to SIKE (supersingular isogeny key encapsulation).

As mentioned in the introduction, there is widespread concern about the possibility of quantum computing; indeed, over the last few years quantum computing has been the big hype, *over*hyped according to some [D,QW]. In this context it is remarkable that, ever since Shor published his algorithm in 1994 [S], researchers predicted that practical quantum computers were 10 years away. Today, almost 30 years later, this prediction essentially has not changed; quantum computers are still a decade away, which makes one wonder whether quantum computers will ever be built at all.

If quantum computing indeed were realized, it would have a devastating effect, invalidating the computational assumptions underlying virtually all widely used public key systems. It would also halve the security levels of popular cryptosystems associated with ad hoc assumptions through Grover's algorithm. The latter can relatively easily be dealt with, but the former will force the introduction of public key systems that have not yet gained the general recognition of the currently still secure systems that we have come to rely upon—indeed, that is the main motivation for Delphi.

Following current terminology, we will use "post quantum" (PQ) for the point in time that sufficiently powerful quantum computers have been developed that weaken current cryptosystems. Quantum issues are further discussed in sections 3 and 4.

3 Computational assumptions

This section discusses, with brief explanations, the two most important and currently widely used computational assumptions, and those that have been selected as the most promising ones for the PQ era. The widely used assumptions are believed to resist traditional computing but will all be invalidated in the PQ era. The PQ ones are believed to resist, many with plausible evidence but all without proofs, traditional computing as well as, again without proofs, quantum computing. As usual all these assumptions may turn out to be wrong from one moment to the next, as most recently happened to the (traditional) computational assumption underlying the PQ candidate SIKE (cf. section 2.4).

Below we do not consider the cryptosystems that can be based on each of the computational assumptions. Whether or not an assumption suffices for the security of either of these cryptosystems is thus also not considered. In other words, RSA, Rabin, Paillier encryption and probabilistic encryption are all lumped together under one assumption; we do not even distinguish between RSA encryption and RSA signatures. This may seem disappointing, but is necessary to keep the number of questions manageable. Depending on the feedback and perceived needs, we may later cast a wider net. Note that soliciting feedback on the security of cryptosystems can hardly ignore how (pseudo)random choices are made, a complication that we prefer not to deal with yet.

3.1 Traditional computational assumptions

Integer factorization is the computational assumption that on a traditional computer it is hard to factor a properly chosen integer (typically the product of two random primes of about the same size). There exists no proof or any type of hard evidence supporting this assumption, except the observation that no efficient general purpose integer factoring algorithm has been published yet. The assumption becomes invalid if a large enough quantum computer were available.

With
$$L[s,c,\gamma] = \exp\left((\gamma + o(1))(\log(s))^c(\log(\log(s)))^{1-c}\right)$$
for $\gamma \in \mathbf{R}_{>0}$, $s \to \infty$ and $0 \leq c \leq 1$, it can heuristically be argued that any integer n can be factored in expected
$$L(n) = L[n, \tfrac{1}{3}, (\tfrac{64}{9})^{1/3}]$$
steps, for $n \to \infty$, using the number field sieve.

Proper understanding of the practical implications of this expression requires experiments. It follows that as a function of the bit length of the supposedly hard to factor number, a rough upper bound for the security level of a cryptosystem based on the hardness of factoring is given by

bit length:	1024	1536	2048	3072	4096	6144	8192	12288	16384
security level:	80	97	110	132	150	179	202	240	270

where it is assumed that close to optimal number field sieve parameter choices are made. If current memory etc. restrictions are taken into account, the security level for bit lengths 1536 and 2048 is closer to 100 and 116, respectively. Informative experiments for bit lengths 3072 or larger require (much) bigger memories than currently available.

Useful lower bounds for the complexity of integer factorization have not been published.

Computing discrete logarithms is the computational assumption that on a traditional computer it is hard to solve the discrete logarithm problem (DL) in a properly chosen cyclic group: given group elements g, h with $h \in \langle g \rangle$ and using multiplicative notation for the group operation, solving DL means finding an integer i such that $h = g^i$. In the generic group model [GGM] the hardness of DL can be proved.

However, the generic group model is not applicable to groups used in practice: for practical purposes there is no proof or any type of hard evidence supporting the hardness assumption, except the observation that for several groups of interest no efficient algorithm that solves DL has been published yet, while for some other groups of interest the assumption was shown to be incorrect. The assumption becomes invalid if a large enough quantum computer were available.

If q is the largest prime divisor of the order of g, then DL can be solved in on the order of $\sqrt{q}$ group operations, using Pollard's rho method. It follows that q must be an at least $2k$-bit prime to get security level k.

It was first suggested by Claus P. Schnorr to work in this smaller (but still large enough) prime order subgroup instead of the full group $\langle g \rangle$. This is more efficient because it allows using smaller exponents; doing so may also be regarded as an additional security risk because those smaller exponents often correspond to "secret" choices in cryptosystems. Below this variant is referred to as "DLS" where "S" refers both to subgroup and to Prof. Schnorr.

The upper bound implied by Pollard's method matches the lower bound for solving DL in generic groups mentioned above.

Below we restrict ourselves to the two types of (sub)groups that are most commonly used.

Finite fields. If g and h belong to (a large enough prime order subgroup of) the multiplicative group of a degree d extension field of a characteristic p finite field then each group element can be represented as a degree less than d polynomial modulo p.

For fixed d this representation can be exploited using a variant of the number field sieve: DL can be solved in expected $L(p^d)$ steps (as for factoring the argument is heuristic). The $o(1)$ in this $L(p^d)$ behaves differently from the $o(1)$ in the earlier factoring-$L(n)$. If $d = 1$ then, as a rule of thumb, if p and n are close so are $L(p)$ and $L(n)$: although they fluctuate around each other for $n, p \to \infty$, for cryptosystems based on this type of DL the same security level estimates can be used (cf. above).

If $d > 1$ care must be taken with $p^d \to \infty$:

- for fixed p (and $d \to \infty$) the computational assumption was recently shown to be wrong (because DL can be solved quite efficiently);
- for fixed d (and $p \to \infty$) the computational assumption is still believed to be valid;
- for p and d both going to ∞ the computational assumption is barely, if at all, used. In this case the situation is murky, with for increasing d and decreasing p a c-value in $L[p^d, c, \gamma]$ that at a given point will be decreasing from $\frac{1}{3}$.

Elliptic curves. For (large enough prime order subgroups of) groups consisting of the points on properly chosen elliptic curves (over finite fields), the way the group elements are in general represented cannot be exploited to solve ECDL (as the elliptic curve DL is commonly referred to, in contrast to DL which from now on refers to DL over the multiplicative group of a finite field): in those groups—when properly chosen, thus avoiding easily recognizable families of known-to-be-weak elliptic curves—-the problem is (believed to be) best solved using Pollard's rho. Because the underlying finite field and the group of points of any elliptic curve over that finite field have cardinalities of the same order of magnitude, security level k for a cryptosystem based on ECDL implies a lower bound $\approx 2^{2k}$ for the cardinality of the finite field over which the elliptic curve is defined.

Remark on DL and ECDL as security assumptions. Irrespective of whether DL or ECDL is used, for security level k the order of the (sub)group $\langle g \rangle$ must have a $2k$-bit or larger prime factor to resist Pollard's rho. Furthermore:

- If DL is used with fixed d, the field cardinality p^d must satisfy $L(p^d) \approx 2^k$ to resist the number field sieve. For $d = 1$ and $k = 80$ or $k = 132$ this leads to a 1024- or 3072-bit (or larger) prime field cardinality. Note that if the order of the generator used has an s-bit largest prime divisor, then s must be at least $L(p^d)^2$, thus $s \geq 2^{2k}$.
- If ECDL is used, the field cardinality must be at least $\approx 2^{2k}$. For $k = 80$ or $k = 132$ this leads to a 160- or 264-bit (or larger) field cardinality.

Note the growing gap between the two minimal field cardinalities for increasing security levels. On the other hand:

- for DL, proper multiplicative (sub)groups of finite fields are easily generated, even on a user-by-user basis, whereas
- for ECDL, generating appropriate elliptic curve (sub)groups is considered to be so challenging that a number of elliptic curve groups has been standardized. This may be considered to be a security issue: not just because attacking a single group affects its many different users, but also because the selection of the standardized groups may not have been as transparent as one would like it to be.

Remark. A difference between integer factorization (IF) and discrete log is which parties makes security-related choices. For IF composites that are supposed to be too hard to factor have to be chosen: this is done independently on a user-by-user basis by each user's associated cryptosystem(s). Whereas for (EC)DL a proper group has to be selected upfront, before the cryptosystem's (EC)DL can be formulated; indeed, in this case it is common practice that the same underlying group is used by many different users. Unless precautions are taken this may lead to advantages for the party that selects the group: some may be skeptical about groups standardized by third parties (cf. comment above).

3.2 PQ computational assumptions

Our choice of post quantum computational assumptions is inspired by cryptosystems that seem to enjoy some level of general support. For none of the assumptions below a functional relationship has been established between parameter choices and security levels: (traditional) experiments have been carried out, but asymptotic results and adequate explanations why certain parameter choices correspond to specific security levels are lacking. With respect to quantum computing resistance—the scenario that would be applicable—these issues are even less clear. Also to avoid biasing the Delphi-contributors while they are considering our set of questions (in Section 5), we prefer to refrain from citing the few inconclusive results that we have come across.

Lattice based. With a lattice defined as a discrete subgroup of $\mathbf{R}^n$, general lattices and ideal lattices are considered. The latter are equivalent to $\mathbf{Z}[X]/(f(X))$ for some monic irreducible polynomial $f(X) \in \mathbf{Z}[X]$ of degree n. They give a square root advantage over full rank general lattices with respect to the size of the lattice description (normally given as a basis over $\mathbf{Z}^n$). This advantage comes at a price: the various NP-hardness results that support, to some extent (i.e., given that their hardness has not further been proved either), computational lattice assumptions so far apply only to general but not to ideal lattices.

The computational lattice assumptions are that the four problems listed below are hard to solve, for general and for ideal lattices, and for "all" common norms; to keep the overall number of assumptions within reason, we limit ourselves to the Euclidean norm. As mentioned above, the "hardness" has not been quantified yet as a function of the lattice parameters, neither using traditional nor using quantum computing. The list below thus represents $2 \times 4 = 8$ computational assumptions, which have to be considered using both traditional and quantum computing:

Shortest vector: find a shortest non-zero element in a lattice.

Gap shortest vector: given real values $\alpha < \beta$, decide if the length of a shortest non-zero element in a lattice is less than α or at least β.

Closest vector: given an element of $\mathbf{Z}^n$, find a lattice element closest to it.

Gap closest vector: given real values $\alpha < \beta$ and an element of $\mathbf{Z}^n$, decide if its distance to the closest lattice element is less than α or at least β.

Coding theory based. Considering just the binary case and integers $0 < k < n$, the 2^k distinct codewords of a general linear (n, k)-code are the binary linear combinations of the rows of a binary $k \times n$ generator matrix G. The problem of decoding a codeword is defined as follows:

Decoding a codeword: given any codeword $c \in (\mathbf{Z}/2\mathbf{Z})^n$, find $x \in (\mathbf{Z}/2\mathbf{Z})^k$ such that

$$xG = c.$$

Codes can be constructed in such a way that decoding is easy by choosing G appropriately. However, lacking knowledge *how* G was constructed, the decoding problem becomes hard, since G then looks indistinguishable from a random matrix. This is the underlying computational assumption in this case; as above, "hardness" has not been properly quantified as a function of k and n. The decoding problem for random linear codes has been proven NP-complete in the worst case—which, as usual, does not give any information about the difficulty to solve any particular instance.

4 Widely used ad hoc assumptions

Each ad hoc assumption below is identified with the cryptosystem that corresponds to it.

4.1 Block ciphers

These algorithms encrypt and decrypt b-bit strings (*blocks*) using a k-bit *symmetric key* (i.e., encryption and decryption use the same key), where b and k are fixed integers. Using a generally accepted chaining method (encryption mode), arbitrary length bit strings can be encrypted and decrypted.

We limit ourselves to the following assumptions:

3K3DES assumption: encryption of a 64-bit block using three successive DES-calls (of which the second is replaced by the inverse DES^{-1} for compatibility with regular DES) with three independently chosen 56-bit keys (for $3 * 56 = 168$ bits of key material) is assumed to have security level 112 (i.e., traditional computational effort of on the order of 2^{112} steps) against decryption without having access to any of the key material. Although this looks (traditionally) secure, the greatest risk of 3K3DES lies in the short block length of 64 bits,

because it makes the method vulnerable to generic collision attacks if on the order of $2^{64/2} = 2^{32}$ blocks are encrypted using a “single” 168-bit key.
In the PQ era the security level supposedly drops to 56.

AES-k assumption for $k \in \{128, 256\}$: encryption of a 128-bit block using AES-k (the *Advanced Encryption Standard*—or simply *Rijndael*—with a k-bit key) is assumed to have security level (close to) k against decryption without having access to the key. Based on Brian Snow’s comment, AES-192 is not considered.
Note the anomaly of the fixed block length of 128 and the alleged security levels: using any generic collision attack, the security level drops to 64 if on the order of $2^{128/2} = 2^{64}$ blocks are encrypted using any fixed k-bit key with $k \in \{128, 192, 256\}$.
In the PQ era the security level of AES-k supposedly drops to $\approx k/2$.

4.2 Stream ciphers

The idea of substituting the random key sequence of the one-time pad by one generated pseudo-randomly is very old, but doing so securely, i.e. constructing strong stream ciphers, has always been a difficult task. Note that one solution is to create a random sequence by using a block cipher in counter mode.

Currently, the two most accepted stream ciphers seem to be Chacha20 (for software implementations) and Trivium (for hardware implementations). As far as we know, there exists no specific quantum attack better than the generic Grover attack against either, so in the quantum setting their security level drops by a factor 2.

4.3 Cryptographic hash functions

These are primitives which quickly provide a fixed length *digest* (or “fingerprint”) of an arbitrary message. A cryptographic hash function with d-bit digests is designed to satisfy the usual three criteria of having security level d (i.e., traditional computational effort of on the order of 2^d steps) against finding pre-images and second pre-images, and security level $d/2$ against finding collisions (a larger security level is impossible due to the birthday paradox). If an input bit flips, each output bit should flip or not with equal probability, but this *avalanche effect* is not commonly explicitly stated as one of the design criteria. Stream-processing of the input and speed are desirable too.

For each cryptographic hash function below the ad hoc assumption is that the hash function’s security levels have the values that it was designed to have. In the PQ era the two pre-image security levels listed above would be halved; the situation with respect to the collision security level is less clear and varies between $d/3$ and $d/2$ depending on the way memory accesses are taken into account.

We limit ourselves to the following four ad hoc assumptions (again leaving out the intermediate security level 192):

SHA2-d assumption for $d \in \{256, 512\}$. Its reliance on the Merkle-Damgård construction makes SHA2, depending on the way it is used, vulnerable to length extension attacks while, on the other hand, maintaining its PQ collision resistance at “$d/2$”. For the rest it seems to be holding up its design promise. It may considered to be an advantage that, since the successful collisions attacks against MD5 and SHA1, it is pretty well understood why SHA2 resists those types of attacks.

SHA3-d assumption for $d \in \{256, 512\}$. SHA3 does not rely on the Merkle-Damgård construction, so length extension attacks do not apply. On the other hand, SHA3 has less extensively been scrutinized than SHA2, seems to be more vulnerable to PQ collision attacks (“$d/3$”), and is slightly slower.

5 Our first set of questions

5.1 General considerations

The actual questionnaire is available on the Delphi website, http://tinyurl.com/delphiquest. Apart from the particular underlying assumptions presented in earlier sections, there are additional considerations which must be taken into account for the questionnaire.

Risk rating Many questions ask you to rate a risk, offering 7 options, from extremely low to extremely high. You can think of this scale as being logarithmic: 0.1%, 1%, 10%, 50%, 90%, 99% and 99.9%. This roughly corresponds to an odd ratio of $1 : 10^k$, with k ranging from -3 to $+3$.

Time span For each question, we expect that our expert considers three different time spans:

- **Short-term** This is essentially a question about the current state of the art, about "now", without taking into consideration possible developments; there are several cases for which the "short term" is omitted because it is irrelevant.
- **Medium-term** In this case we ask our expert to consider the likelihood of improvements during the next 10 years.
- **Long-term** Same as the previous item, but for a period of 40 years.

Self assessment For each question a box can be ticked to indicate one's level of expertise in the field at hand and one's level of confidence in the response given.

Remarks and possible conflicts of interest Remarks of any kind can be entered too, in free format, in comment boxes—these would, among others, be a good place to disclose conflicts of interest.

Security level As mentioned above, we have a reasonable notion of what adequate security means relative to current academic attacks—and we know Brian Snow's opinion. In the first question we ask an estimate of rigorous security against attacks sponsored by well-funded state actors.

Question 1 (SL) *What is the smallest k below for which you find that it is infeasible for a well-funded state actor to conduct a brute force search of 2^k alternatives? Choose $k \in \{80, 96, 112, 128, 160, 192, 224, 256\}$.*

Here, and below, the term "infeasible" must be interpreted at face value, i.e., that it cannot be carried out even by the algorithmically and mathematically most sophisticated and computationally most capable party.
This same question is repeated considering time span of 10 years, and of 40 years.

Quantum computing There is no consensus on when or whether quantum computing will have a practical impact on cryptography. We deal with this issue by including the following question:

Question 2 (QC1) *Do you believe that Quantum Computing will constitute a realistic threat to some cryptographic primitive in the next 40 years? Choose YES or NO.*

This question is used to tailor the remaining questions shown: in the case that quantum computing is considered possible (i.e., the expert's answer is YES), then each question below will have two variants: one referring to a traditional computer, and one to a quantum computer. The second variant will be suppressed if quantum computing is not considered a possibility.

Question 3 (QC2) *How do you rate the risk that problems that are currently too hard to be solved on traditional computers can be solved using one or more actual quantum computers? Please give answers for the medium-term (up to 10 years) and long-term (up to 40 years)*

For the short term the answer is, taking only published information into account, known to be 0%.

We now list the questions related to the various assumptions introduced earlier. To avoid tedious repetition we state them for the short term and using a traditional computer; we do not state the variants obtained by medium and long term, and a quantum computer (for which short term is omitted).

5.2 Traditional computational assumptions

Question 4 (IF and IF-QC) *For the bit length choices k listed below, define IF-k as the computational assumption that on a traditional/quantum computer it is infeasible to factor a properly chosen integer of k bits. Which is, in your opinion, the smallest value of k to guarantee that IF-k is infeasible? If you believe no such value exists, choose infinity. Choose $k \in \{1024, 1536, 2048, 3072, 4096, 6144, 8192, 12288, 16384, \infty\}$.*

Question 5 (DL and DL-QC) *For the bit length choices k listed below, define DL-k as the computational assumption that on a traditional/quantum computer it is infeasible to compute the discrete logarithm in the full multiplicative group Z_p^* where p is a k-bit prime for which p?1 has a prime factor of order at least $L(p)^2$. Considering the short term, which is, in your opinion, the smallest value of k to guarantee that DL-k is infeasible? If you believe no such value exists, choose infinity. Choose $k \in \{1024, 1536, 2048, 3072, 4096, 6144, 8192, 12288, 16384, \infty\}$.*

Question 6 (DLS and DLS-QC) *For the bit length choices k listed below, define DLS-k as the computational assumption that on a traditional/quantum computer it is infeasible to compute the discrete logarithm with respect to an element of k-bit prime order in the group Z_p^*, where p is a n-bit prime for n such that the DL-n assumption holds. Which is, in your opinion, the smallest value of k to guarantee that DLS-k is infeasible? If you believe no such value exists, choose infinity. Choose $s \in \{160, 192, 256, 384, 512, \infty\}$.*

Question 7 (ECDL and ECDL-QC) *For the bit length choices k listed below, define ECDL-k as the computational assumption that on a traditional/quantum computer it is infeasible to compute the discrete logarithm with respect to a element of k-bit prime order in the group of a (not known to be weak) elliptic curve over an order k-bit prime field. Considering the short term, which is, in your opinion, the smallest value of k to guarantee that ECDL-k is infeasible? If you believe no such value exists, choose infinity. Choose $k \in \{160, 192, 256, 384, 512, \infty\}$,*

Question 8 (EC Standards) *For the value of k you selected in the previous question, do you feel comfortable using standardized ECDL group choices such as those proposed by NIST, Certicom, BSI (Germany), Daniel J. Bernstein, or others? Please comment.*

5.3 Block and stream cipher cryptanalysis

The questions below ask for a risk evaluation for the five ciphers mentioned above (3K3DES, AES-{128, 256}, Trivium, and Chacha20) for the medium and long term; for short term the (published) risk is known to be 0%.

Question 9 (3K3DES) *How do you rate the risk that the security level of the block cipher 3K3DES will be reduced by 10% or more (i.e. from 112 to 101 bits or less) when using traditional computing? And when using quantum computing?*

Question 10 (AES-128) *Same question, for AES-128 (i.e., from 128 to 115 bits).*

Question 11 (AES-256) *Same question, for AES-256 (i.e., from 256 to 230 bits).*

Question 12 (Trivium) *Same question, for Trivium (i.e., from 80 to 72 bits).*

Question 13 (Chacha20) *Same question, for Chacha20 (i.e., from 256 to 230 bits).*

5.4 Cryptographic hash function cryptanalysis

Question 14 (SHA2 $\leftrightarrow$ SHA3) *How do you compare the security of SHA2-256 and SHA3-256 on a scale from 0 to 10, where 1 and 9 stand for "a lot", while 3 and 7 stand for "somewhat", and 5 for "indifferent"?*

Question 15 (SHA2-256) *How do you rate the risk that a collision will be found for the hash function SHA2-256 when traditional computing is used? And when quantum computing is used?*

The next three questions are identical to the last question above, but respectively refer to SHA2-512 and SHA3-{256,512} instead.

5.5 Post quantum assumptions

As explained before, we have 4 problems with 2 types of lattices, leading to 8 different questions, plus a single question about decoding linear codes. We use "(G)SV" for (gap) shortest vector, "(G)CV" for (gap) closest vector, "-G" or "-I" for "general" or "ideal" lattices, and "DRLC" for the decoding problem for a random linear code. Please see reference [NIST] for current parameter choices for each desired security level, and for all 9 problems considered in this section. The somewhat vague formulation of the questions corresponds to the current uncertainty how, for each of the problems under consideration, the security level behaves as a function of the parameters used.

Question 19 (SV-G) *How do you rate the risk that cryptanalysis of problem SV-GL will render current parameter choices ineffective for either of the desired security levels, with choices and levels as presented in NIST IR 8413 (https://csrc.nist.gov/pubs/ir/8413/upd1/final), when traditional computing is used? And when quantum computing is used?*

The next seven questions are identical but respectively refer to SV-I, GSV-{G,I}, (G)CV-{G,I}, and DRLC instead.

6 Reporting the results

All responses received will be made accessible on the Delphi website, ideally non-anonymous (but depending on the contributor's preference). Summaries will be made available on the website and in updates to this paper. At this point in time there is nothing to report yet.

Unanimous responses can easily be dealt with, but unanimity is not what we expect. Most likely we will have to report, one way or another, non-unanimous responses, majority opinions, and various "average" opinions depending on the wide variety of weighing factors that can or may be used: none, based on "seniority" (h-factor, number of papers at IACR conferences, ...) confidence level, expertise level, ...

7 Concluding remarks

Our interests and main expertise is in some of the practical aspects of cryptography, as reflected in this paper. Thus our approach is more pragmatic and hands-on than papers such as [GK, JF].

There are many other areas, in (more theoretical) cryptography, complexity theory, theoretical computer science in general, and even mathematics, that we are much less versed in but where a similar collection of opinions could be informative. Despite our attempt to limit our questions to just the "most important" ones, we feel that the size of our questionnaire already got way out of hand: even covering just our own field more completely will be much worse; we do not want to think about the avalanche of questions once more fields, as above, are included as well. It is not that collecting such data would give any results that may be relevant for those fields, but it may give an indication in which direction trends are moving or switching—and we may find out if there is any agreement on P versus NP and which of Impagliazzo's five worlds we live in, to mention two of many, many examples of "popular" questions. If this is just for entertainment or would serve a "real" purpose remains to be seen.

We do not deny that one of the reasons behind this paper is simply to satisfy our own curiosity—and to find out how many of our colleagues follow the lead of their fomo-riddled deans and university presidents. It also has the possibly positive effect, as mentioned in the introduction, that it may help crypto-decision-makers to make up their minds, hopefully in a good direction: the suggestions implied by our paper may eventually turn out to be wrong—due to circumstances unforeseen by the average subject matter expert—which may lead to "damages" or have other undesirable consequences. If so, the mere fact that those misleading suggestions were properly justified and documented may still offer useful protection to the parties involved.

All information made available through Delphi is provided in good faith. But obviously, the authors do not take any responsibility for any of the consequences of following (or not) any of the recommendations that may be inferred from the paper or web site.

Acknowledgment The second author fondly remembers his conversations with Brian Snow (who sadly passed away in December 2022), and thanks his former colleagues Aymeric Genêt, Novak Kaluderovic and Thorsten Kleinjung for many useful discussions and valuable insights gained during their weekly lunches at EPFL.

References

BDRSSTW1996 M. Blaze, W. Diffie, R.L. Rivest, B. Schneier, T. Shimomura, E. Thompson, M. Wiener, Minimal key lengths for symmetric ciphers to provide adequate commercial security, www.bsa.org/policy/encryption/cryptographers-c.html, January 1996.

BS https://en.wikipedia.org/wiki/Brian_Snow.

BSPC Personal communication of the second author with Brian Snow, June 2014.

D M. Dyakonov, The case against quantum computing, IEEE Spectrum, 15 Nov. 2018.

GGM https://en.wikipedia.org/wiki/Generic_group_model.

GK S. Goldwasser and Y. T. Kalai, Cryptographic Assumptions: A Position Paper. TCC 2016. https://eprint.iacr.org/2015/907.

JF S. Judson, J. Feigenbaum, On Heuristic Models, Assumptions, and Parameters https://arxiv.org/pdf/2201.07413.pdf.

KM N. Koblitz and A. Menezes, Another Look at "Provable Security", J. of Cryptology 20 (2007) 3–37.

LV2001 A.K. Lenstra, E.R. Verheul, Selecting cryptographic key sizes, J. of Cryptology 14 (2001) 255–293.

NIST https://csrc.nist.gov/pubs/ir/8413/upd1/final

S P.W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, FOCS 1994.

QW https://www.youtube.com/watch?v=CBLVtCYHVO8.