Joint Interpretation Library

Management of

Code Disclosure and Software IP Reuse

CARRYING OUT IMPACT AND VULNERABILITY ANALYSIS
IN CASE OF
REUSE OF SOFTWARE IP
DISCLOSURE OF SOFTWARE AND FIRMWARE

Version 1.2
November 2017

This page is intentionally left blank.

Table of content

1 Foreword 4
2 Introduction 5
3 IP Reuse during Life Cycle phases 1 and 2 7
3.1 Recommendation 9
4 Disclosure of Code of already certified products 10
4.1 Actors and first action in case of disclosing 10
4.2 Analysis of disclosing cases 10
4.3 Focused on Security 12
4.3.1 Case 1: Firmware disclosed – no security relevant secrets contained 12
4.3.2 Case 2: Firmware disclosed – with security relevant secrets 12
4.3.3 Case 3: Software without risk potential – OS and Application - disclosed 13
4.3.4 Case 4: Software with risk potential – OS and application, no cryptographic software - disclosed 13
4.3.5 Case 5: Software – OS and application, with cryptographic secrets – disclosed 13
4.3.6 Case 6: Cryptographic Software of a public algorithm disclosed without protective implementation 14
4.3.7 Case 7: Cryptographic Software of a public algorithm disclosed with protective implementation 14
4.3.8 Case 8: Cryptographic Software of a proprietary algorithm disclosed 15
4.4 Focused on IP protection 15
4.4.1 Cases 1 and 2: Firmware disclosed 15
4.4.2 Cases 3, 4 and 5: Software disclosed 16
4.4.3 Cases 6 and 7: Cryptographic Software of a public algorithm disclosed 16
4.4.4 Case 8: Keeping a proprietary algorithm as an asset 16
Overview Table 17
5 Conclusion 18
6 References and Abbreviations 19

1 Foreword

1 This is a supporting and public document, intended to give the Common Criteria using and participating parties a general guideline how to evaluate the reuse code and also how to deal with published code.

2 Supporting documents may be “Guidance Documents” that highlight specific approaches and applications to areas where no mutual recognition of its application is required, and as such, are not of normative nature, or “Mandatory Technical Document”, whose application is mandatory for evaluations whose scope is covered by that of the supporting document. The usage of the latter class is not only mandatory, but certificates issued as a result of their application are recognized under the SOG-IS.

2 Introduction

3 Recent attack scenarios have raised the need to discuss rules and recommendations about how to handle the emerging risks and security affects – if any – where firmware (FW) or software (SW) code, or parts of it, are disclosed or published. In general there are two ways how code may become public in the sense of certification: One is in the real course of an attack and the second way is the reuse of public available code. The reuse case can also split up into the reuse of public available code parts and the reuse of code parts kept confidential but used also on other non-certified products.

4 The use of public sources during development for a certified product is not covered by the protection profile [1]. In principle, with [1], it must be distinguished whether the published code implements a threat to the certified system – consisting of hardware, IC dedicated firmware and user software or data – or not. If formal consequences are to be applied shall be a result of analyzes of the claims 60, 61 and 62 of the PP [1] . Here is the relevant citation from [1]:

> *60 The user (consumer) of the TOE places value upon the assets related to high-level security concerns:*
>
> *SC1 integrity of user data of the Composite TOE,*
>
> *SC2 confidentiality of user data of the Composite TOE being stored in the TOE's protected memory areas,*
>
> *SC3 correct operation of the security services provided by the TOE for the Security IC Embedded Software.*
>
> *Note the Security IC Embedded Software is user data and shall be protected while being executed/processed and while being stored in the TOE's protected memories.*
>
> *61 The Security IC may not distinguish between user data which is public knowledge or kept confidential. Therefore the security IC shall protect the user data of the Composite TOE in integrity and in confidentiality if stored in protected memory areas, unless the Security IC Embedded Software chooses to disclose or modify it.*
>
> *62 In particular integrity of the Security IC Embedded Software means that it is correctly being executed which includes the correct operation of the TOE's functionality. Parts of the Security IC Embedded Software which do not contain secret data or security critical source code may not require protection from being disclosed. Other parts of the Security IC Embedded Software may need to be kept confidential since specific implementation details may assist an attacker.*

5 The PP [1] mitigates the formal position of the forerunner PP0035 where any type of code disclosure, including also the use of code from public sources, could lead to a formal problem.

6 For the common understanding it is important to note that this document differentiates between code originated by the hardware developer as firmware and code developed by the user as operating system and/or application. These two types of code lead usually to two types of TOE in two different certification processes. In the following, differentiated firmware and software cases occur, but also general comments where code as such is addressed. The term code as such addresses in these cases both, the hardware developer firmware and the user developed software.

7 The disclosure of code may not make the TOE significantly more vulnerable to an attack on the operational assets such as stored value. For example, if the published code is only I/O routines, then such code is generally not critical for the security of the TOE and its public knowledge does not increase its vulnerabilities. Indeed, in practice it may

not be hard for an attacker to deduce implementation detail for IO routines, since their nature is often similar from product to product. Nevertheless, the designation of code as uncritical in this sense shall of course be within the scope of the evaluation.

8 However, the public presence of code and also the reuse cases in general raise questions, including

- How to react, if the published piece of code has been reused on various products by different developers and if this code would be critical for security?
- How to deal with the case where developers take standard code from public sources to implement security critical functions? This could be for example for published cryptographic algorithms and would be a matter of degree of modification and also protection for example by the hardware.
- This list is non-exhaustive, other cases could occur.

These scenarios would jeopardize the required level of resistance to attackers during an evaluation, as well as the surveillance/reassessment process of certificates claiming strict conformance to [1].

9 Instead it is recommended that if the source code becomes public, then a reassessment of all attack paths may become necessary, depending on the type of disclosed information. And, such a reassessment may apply to all products sharing this piece of code. The reassessment will determine whether the piece of disclosed information is security critical or not and therefore whether or not this has consequences for the certification status.

10 On the other hand, if a developer implements code from public sources then is should be made known to and considered by the evaluator.

3 IP Reuse during Life Cycle phases 1 and 2

11 During the development phases 1 and 2 – referred to [1] – it is necessary for economic and practical reasons – such as incorporating state of the art features - to reuse code parts in firmware and software.
In the following we separate code between implementation in certified and non-certified products. If code is used in certified products, it is seen here per definition as used solely in these certified products and not in non-certified products. It is also the case that not all products used in security applications have undergone a certification process; nevertheless these products could contain security relevant secrets and assets to be protected from developers view.
This view results in the following scenarios:

Reuse of code parts originating from Proprietary Sources	
1. Code used in certified products	**2. Code used in non-certified products**
a. Kept confidential for security reasons	a. Kept confidential for security reasons
b. Kept confidential for IP protection reasons	b. Kept confidential for IP protection reasons (1)
c. Developed in secure and audited environment as required by PP0084	c. Alternative possibilities: i. Developed in secure and MSSR audited environment ii. Developed in unprotected environment or by third parties with/without NDA
Reuse of code parts originating from Public Sources	
3. Used in certified products	**4. Used in non-certified products**
a. Not confidential due to public source.	a. Not confidential due to public source.

(1) As long as no formal proof of the level of confidentiality is given, i.e. ALC_DVS, the code has most likely to be rated as not confidential or even public.

12 These scenarios could result in following:

- As long as we stay entirely in 1. a to c we are conform to [1] and there is no further need for analysis or declaration during development.
- If we mix up 1.a to 1.c with any item 2.a to 2.c, then the impact of code disclosure on the individual case has to be considered with regard for what, in the sense of security critical purpose, how and also what amount of reused code is disclosed. This is because we have to consider the worst case for what an attacker could learn by comparison between unprotected and protected devices on which the code is reused. For example, this may expose points at which the different devices behave in the same way, and may therefore offer opportunities for the attacker to learn from the unprotected device how to manipulate certain aspects of the protected device. This could lead to an exploit on a protected and certified device. Therefore, from the attacker's perspective it does not matter, whether the development is done in the protected environment 2.c.i or not 2.c.ii.
- In that case, meaning the mix-up 1.a to 1.c with any item 2.a to 2.c, the factors for elapsed time, expertise or knowledge are affected and shall be subject of the risk assessment of the developer and evaluation of the ITSEF.
- Nevertheless, if the cases 2.a or 2.b are given in conjunction with 2.c.i, and also mixed up with 1.a to 1.c points for knowledge could be given, as the attacker

does not have the source code. This holds true as long as the formal prove of the level of confidentiality is given. The efforts for learning on unprotected devices and developing an exploit for the targeted protected devices must be considered and a rationale is required.

- If 2.c.ii holds true, even for parts of it, the developed code carries a permanent risk for disclosure at any time – regardless whether a publishing or other disclosure already happened or not. Due to changes in staff the NDA may protect a while, but one cannot rely on it in general. The risk depends then on the details of the individual NDA. For example, one NDA applies only as long as a person is member of the developer company, others last for years after the person has left the company. As such NDAs could not be subject of the evaluation and as a consequence, the risk depends generally whether critical or uncritical code were subject of the third party development.
  From the certification evaluation perspective this code is seen as public.
- In case of critical code parts in third hands, no points can be given for knowledge. Whether other factors such as elapsed time, expertise or equipment are affected shall be subject of the risk assessment and analysis of the ITSEF. A rationale is required.
- The use of 3 always requires risk assessment and analysis. Consequently, the risk depends whether critical[1] or uncritical code origin from public sources. In any case no points can be given for knowledge. Whether other factors such as elapsed time, expertise, access to TOE or equipment are affected shall be subject of the risk assessment and analysis of the ITSEF.
  It must be considered that the development of an attack path could be enabled or simplified, if the equal code is used on certified secured, but also on unsecured chips – i.e. freely available elsewhere. A rationale is required.
- Disclosure of code used in scenario 4 does not affect certified products.

[1] The term critical is meant her in the context of security protection.

3.1 Recommendation

13 For preparation of the evaluation and simplification of the risk assessment during the evaluation process, it is recommended for the developers to identify and mark the code parts reused from other sources – and name their origin - in order to provide the ITSEF a clear picture. This holds for the two cases:

- Code development with reuse of code parts whereas the development result is subject of certification and
- Re-use of already certified code in other for example non-security products, where the target is not to jeopardize the required level of resistance to attackers during an evaluation, as well as the surveillance/reassessment[2] process of certificates claiming strict conformance to [1].

14 In accordance with [1] it is subject of the analysis and assessment whether a marked piece of code remains as asset or if it can be dropped from the asset list. For example simple input/output routines[3] could be dropped from the asset list, but, if a piece of firmware/software has been identified as being critical, it earns the attribute asset in the sense of [1]. Consequently, such code must be kept confidential and conformant to the assurance classes related to the life cycle support.

15 It is important for common understanding that a piece of code - declared as uncritical and not being an asset - remains part of the evaluation and assessment. It is not excluded from the process as the evaluation and analysis coverage shall remain complete.

[2] The naming depends on the scheme.

[3] For clarification reasons please note that this example does assign input/output routines as uncritical in principle: In the past badly implement interface routines have been exploited for example for memory dumps. Therefore, these parts remain of course subject of the evaluation.

4 Disclosure of Code of already certified products

4.1 Actors and first action in case of disclosing

16 It is part of the "nature" of code disclosing or attack descriptions or recipes how to break something that no one can limit or control where, when, how and to whom this happens. In the worst case, code and other relevant details are published on internet platforms and are immediately available worldwide.

17 Those sources are principally available to anybody and therefore each of the involved parties (i.e. Certification Body, ITSEF, hardware developer or software developer) could be the first to discover such case.

18 If such case is given, and the publication content jeopardizes the security of a product, fundamentals of the given certificate are in question and the discoverer shall inform the other parties involved i.e. Certification Body, ITSEF, hardware developer and software developer. It is recommended to handle this rule as mandatory obligation for all certification involved parties.

19 It is of course in the interest of the injured vendor to limit possible consequential damage to their customers/users and by that these parties will actively urge an own reassessment, as soon as they are informed of a disclosure – even to distinguish whether it is critical or not. Nevertheless, such reassessment shall not be on a voluntary basis and the following rules shall apply:

- The affected developer is obligated to conduct a first reassessment clarifying whether critical or uncritical code disclosure has happened. The result and rationale must be communicated to Certification Body and ITSEF.
- If it is considered as a critical case, a reassessment involving the ITSEF is mandatory resulting in an evaluation testing report and rationale.
- If the first vendor reassessment and the ITSEF reassessment does not take place in appropriate short time, the Certification Body takes the appropriate action in order to minimize the possible risks for the users. More precisely, the Certification Body may inform other parties about the vulnerability in case of surveillance/reassessment process.

4.2 Analysis of disclosing cases

20 In the following we foresee a non-exhaustive list of scenarios where we split up the software into firmware (FW), operating system (OS) and application (AP), as well as into pure security aspects and IP-aspects.

21 For clarification, note that the FW is a piece of software running on the hardware, usually used for basic chip configurations at start-up and to provide an interface between hardware related functions and the layers of OS and AP on top of it.

22 In general, the FW rarely contains true secrets and is rather a target for perturbation attacks in order to manipulate chip configurations or access other operation modes than normal users have. A disclosure of FW is therefore rather a topic in terms of IP-rights. On the other hand a disclosed FW could make it easier for an attacker to develop and mount several kinds of attacks which could be exploitable in a later stage. Therefore, we cannot say in general that FW can be dropped from the asset list.

23 But, it has also to be considered that only parts of FW, OS or SW contain secrets. From a pure security perspective, a disclosure of a product-specific secret declared as asset, usually does not affect the rest of code used in other products.

24 Nevertheless, in case even one FW, OS or SW part is assigned to be a security asset, the

hardware must provide appropriate countermeasures against the readout and unintended modification of this specific asset. If, despite all actions taken, the disclosure of FW, OS or SW parts occurs, the vendor must conduct an assessment to determine whether critical or uncritical parts have been disclosed. In the critical case, the certificate(s) is/are subject of a mandatory risk analysis by the ITSEF - following the rules proposed in chapter 4.1.

25 The risk analysis should differentiate between different cases depending on the hardware:

- If a SW/FW asset on a certain product A has been disclosed the FW, OS or SW still could be secure on another, different product B providing sufficient countermeasures in software and/or hardware against readout.
- Also, if the other chip B uses the disclosed software but implements different physical structure, timing, different encryption / keys and/or technology, it could still be not practical to find an exploitable vulnerability.
- And even more, if there is another chip B equal in hardware to chip A using the disclosed code but implemented in a different application or OS, it could still be difficult to find an exploitable vulnerability.

26 In contrast to that an attack could be exploited in a simplified way, if, for example, a secret master key has been disclosed, or if the disclosed software provides unintended triggers or other weaknesses for subsequent analysis leading directly to successful attack paths.

27 These examples underline that a risk analysis is essential to analyze the detailed situation if critical FW, OS or SW has been disclosed. And, the analyst should bear in mind that an attacker may simply attempt an attack without first determining whether the compromised asset is present on the new target chip or not – simply by trial and error.

28 In case of disclosure of uncritical code, or when using public code to process uncritical data, then it is unlikely that there is impact on security and certification. But, when processing critical assets, code or hardware must provide appropriate countermeasures for those operations and may apply therefore only suitable modified public code, in order to complicate its identification, or use real proprietary alternatives. It is of importance for developers to consider that the reuse of public code and even standards can implement inherent weaknesses, and, these weaknesses are even sometimes publicly discussed and can be countered with the implementation. Nevertheless, developers should pay attention and analyze public sources very well since there is no guarantee that inherent weaknesses of public sources are already known, discussed and published.
The evaluation shall consider all scenarios and risks.

4.3 Focused on Security

29 In the sections below we identify disclosure scenarios and their associated assumptions. For example, we distinguish the cases where knowledge of source code assists in making a successful perturbation attack (case 4) and where it does not (case 3). The assumptions in each case lead to conclusions about whether the FW, OS or SW should be treated as an asset and about the effect of its disclosure on a certified product that contains it.

4.3.1 Case 1: Firmware disclosed – no security relevant secrets contained

30 This is the case where we assume the disclosure of the firmware which is used to apply the chip configuration during start-up. At that point no user software is involved. As no secrets are handled that time, the only target for an attacker would be achieving an unintended, from security point of view, weaker hardware configuration. Such configuration could be for example keeping a specific hardware security mechanism disabled.

31 A state of the art startup procedure is as such critical for subsequent chip operation and is therefore usually protected by special hardware means. Perturbation or modification trials during start-up will therefore not lead to an exploitable situation being present at the moment the user software takes over, but to a chip reset and start over.

32 Nevertheless, this startup firmware is and remains also part of the evaluation, as it must be considered, if this disclosed firmware would enable or simplify the mounting of a perturbation attack on a later stage in operation.

33 **Asset status**: If the evaluation or assessment does not encounter risks during startup and risks for simplified mounting of a perturbation attack at a later operating state, this specific firmware part could be dropped from the asset list.

34 **Impact on certifications**: An evaluation or surveillance/reassessment process should not be affected, but a rational in an update of the ETR-composition could be required.

4.3.2 Case 2: Firmware disclosed – with security relevant secrets

35 Assume the hardware vendor has implemented in the firmware critical secrets. These could be for instance a proprietary software algorithm or means of how a secret key is built or, even worse, the secret key is stored in fixed form.

36 In that case, even a change of the keys or the parameters of the secret FW part does not really help, as the published source code would allow for analysis of when does what happen and when is the secret used or built. This opens for side channel analysis and also well timed perturbation attacks, which we assume is significantly simplified by having the source published. It has to be considered not only when, but also "where" and "how" this happened. Source code analysis would allow identifying where this secret is being stored in memory and how it is computed.

For instance assuming access protecting secrets (PIN) are stored enciphered in the memory by SW, the FW or OS may hold an ICC key being used for enciphering these secrets. If the attacker gets a memory dump of the TOE, he can therefore more easily recover the ICC key value. Even if the key is different on each TOE, knowing the location may help for exploitation. In addition, if the use of the ICC key has custom parameters (e.g. specific value in Triple-DES CBC encryption), knowing how the ICC key is used also may help to recover the values of AP secrets. Therefore, a rewriting of the entire code or other deep modifications including the change of secrets/keys could help to maintain security on products, making use of these modified code parts.

37 **Asset status**: The secret part of the firmware was indeed an asset which had to be protected but is now public. All products using the unchanged, published no more secret

parts of the code are affected. The confidentiality of the asset(s) is compromised and leads to an exploitable vulnerability.

38 **Impact on certifications**: Risk analysis of the attack path(s) by the ITSEF/Certification Body can therefore lead to a recalculation of points and/or consequently impacts the required level of resistance to attackers during an evaluation or surveillance/reassessment process. Even if the risk analysis results in no change to the certification, a rationale in an update of the ETR-composition could be required.

4.3.3 Case 3: Software without risk potential – OS and Application - disclosed

39 If we assume that the real secrets like secret keys (of any kind) and proprietary algorithms are not disclosed and are handled properly in a secure chip, we also find no clue what really would make a difference for an attacker if the OS and application are public. A public software specification or source code would give some hints when does what happen, but this is not a very worthy information as the really interesting things, for example when does the cryptographic coprocessor compute something with the secret key, could be analyzed by power profiles. The public availability of the source code would just reduce the time required for the power profile analysis. Public availability would also be no lack of security, if only the keys and secret functions are protected by means of an appropriate encryption.

40 We consider this case also as given, if a TOE comes with the implementation of a cryptographic service not being declared as security functional requirement SFR or in context with security policy/ies. In this case the evaluation focusses just on functional verification and not on security evaluation, and in addition, it is then regardless, from certification point of view, if the code is public or not. It still could matter the IP side.

41 **Asset status**: From a pure security point of view, the software could be dropped from the asset list.

42 **Impact on certifications**: An evaluation or surveillance respectively reassessment should not be affected.

4.3.4 Case 4: Software with risk potential – OS and application, no cryptographic software - disclosed

43 In general, if a published source code (firmware or software) would not contain secrets, as in case 3, but would allow to analyze other vulnerabilities, for example the identification of a dedicated point in time useful for a perturbation in order to achieve special operating modes or to reset security relevant counters or similar, then it still has to be considered as a disclosed asset. In these cases it could be quite hard to identify such critical coding and the impact analysis could be more complex.

44 **Asset status**: Therefore, it is justified that the identified SW code parts opening vulnerabilities for subsequent attacks steps shall be assigned as asset. Consequently they have to be protected. All products using the unchanged, published and no more secret parts of the SW are affected.

45 **Impact on certifications**: The confidentiality of the asset(s) is compromised and leads to an exploitable vulnerability. Risk analysis of the attack path(s) by the ITSEF/Certification Body can therefore lead to a recalculation of points and/or consequently impacts the required level of resistance to attackers during an evaluation or surveillance/reassessment process. In that case, even if the risk analysis results in no change to the assessment result, an update of the ETR-composition is required.

4.3.5 Case 5: Software – OS and application, with cryptographic secrets – disclosed

46 Here we have the case that the OS or application developer implemented secret keys (of any kind) or means of how they are built - using for instance a proprietary algorithm - in the software. Again, usually the major part of the software is not security relevant and

we have to consider two parts: a secret one and one which could be disclosed without security relevant effect. Note that the uncritical part even does not open vulnerabilities for subsequent attack steps.

47 Even a change of the keys or the parameters of the secret code part does not really help, as the published source code would allow for analysis of when does what happen and when is the secret used or build. This opens for side channel analysis and also well timed perturbation attacks, which we assume is significantly simplified by having the source published. Therefore, a rewriting of the entire code or other deep modifications including the change of secrets/keys could help to maintain security on products, making use of these modified code parts. However, such modification always requires a reassessment/recertification by the ITSEF and certification body.

48 **Asset status**: The secret part of the operating system or application was indeed an asset which had to be protected but is now public. All products using the unchanged, published parts of the SW (since they are no longer secret) are affected.

49 **Impact on certifications**: Risk analysis of the attack path(s) by the ITSEF/Certification Body can therefore lead to a recalculation of points and/or consequently impacts the required level of resistance to attackers during an evaluation or surveillance/reassessment process. In that case, even if the risk analysis results in no change to the assessment result, an update of the ETR-composition is required.

4.3.6 Case 6: Cryptographic Software of a public algorithm disclosed without protective implementation

50 In this case the software developer, of a hardware or software vendor, programmed straight down from public sources and did not implement any countermeasures against side channel, perturbation or physical attacks. By this, it is also not relevant whether a secret key has been handled properly by the chip – we assume that any certified TOE provided sufficient resistance to attacks because of other measures implemented in the IC or (non-disclosed) software, or else that the public information was not used for any security relevant purpose.

51 **Asset status**: From a pure security point of view, the software could be dropped from the asset list, since it is an implementation of public sources.

52 **Impact on certification**: An evaluation or surveillance/reassessment process should not be affected. However, an update of the ETR-composition could be required, in order to note that the details of the algorithm implementation must now be considered public knowledge.

4.3.7 Case 7: Cryptographic Software of a public algorithm disclosed with protective implementation

53 This appears to be different compared to the prior case, as the implementation of countermeasures against side channel and perturbation attacks are the assets. This disclosure could allow for a more detailed timing analysis when for example a secret key is used. Significant simplification of side channel attack and/or perturbation scenarios, as well as published attack recipes can be expected and therefore this should be considered as being an asset.

54 Even a change of the keys or the parameters of the secret SW part does not really help, as the published source code would allow for analysis of when does what happen and when is the secret used or build. Therefore, a rewriting of the entire code or other deep modifications including the change of secrets/keys could help to maintain security on products, making use of these modified code parts.

55 **Asset status**: From a pure security point of view the secret parts of cryptographic software of a public algorithm should be kept as an asset for the case it enables an

attack. All products using the unchanged, published parts of the SW (since they are no longer secret) are affected.

56 **Impact on certifications**: Risk analyses of the attack path(s) by the ITSEF/Certification Body can therefore lead to a recalculation of points and/or consequently impacts the required level of resistance to attackers during an evaluation or surveillance/reassessment process. In that case, even if the risk analysis results in no change to the assessment result, an update of the ETR-composition is required.

4.3.8 Case 8: Cryptographic Software of a proprietary algorithm disclosed

57 Proprietary algorithms are implemented to complicate cryptographic analysis of an attacker significantly, as there are not public sources and discussion of the algorithm available. Attackers do hard to analyze such algorithm is offered to the OS or AP as black box. Disclosure of a proprietary algorithm would be expected to have a significant impact on the resistance to attack.

58 **Asset status**: From the pure security point of view the Cryptographic Software of a proprietary algorithm should be kept as an asset. The confidentiality of the asset(s) is compromised and leads to an exploitable vulnerability.

59 **Impact on certifications**: It could be very complex, time consuming and expensive analyzing the impact of a disclosed proprietary algorithm. And, there is also some feasibility that a fundamental flaw – if present - in the algorithm could not be identified during the limited time of the generation of the impact analysis. In order to limit the risk, although we assume that the analysis could not be finalized in all details, the ITSEF/Certification Body analysis can lead to a recalculation of points and/or consequently impacts the required level of resistance to attackers during an evaluation or surveillance/reassessment process surveillance/reassessment process. In that case, even if the risk analysis results in no change to the assessment result, an update of the ETR-composition is required.

4.4 Focused on IP protection

60 The following notes have been introduced to clarify the difference between intellectual property assets (IP) and security relevant assets. Various discussions have concluded that it is important for achieving a common understanding that IP assets are seen different from the security assets discussed above. Indeed, in some cases a disclosed IP asset could have no effect on security of the product, but on the other hand, an implemented security feature could also be an IP asset.

61 Because of the potential impact on certified products of disclosure of IP assets that are also security assets, we argue that a more detailed definition of the embedded software aspects should be given. The following notes attempt to mitigate potential effects of future disclosures of IP assets. The notes refer to classification of code as an “IP-asset” to distinguish this case from the security assets discussed above.

62 The following notes have a recommendation and not a regulation character. It is of course up to the owners and developers how they manage their IP assets.

4.4.1 Cases 1 and 2: Firmware disclosed

63 The firmware is usually targeting a specific hardware platform and cannot be used elsewhere. It could give some hints of what is solved how but in general no critical IP is included. From that perspective it could be disclosed without harm. However, we can in general not exclude that IP needing protection is included and therefore, the case that the firmware is still an asset can occur.

64 From pure IP point of view the firmware or even parts of it could be optionally assigned

to be an IP-asset.

4.4.2 Cases 3, 4 and 5: Software disclosed

65 Depending on the software vendor and whether IP - that requires protection - is included, it is an option to be kept as an asset.
From pure IP point of view the software or even parts of it could be optionally assigned to be IP-asset.

4.4.3 Cases 6 and 7: Cryptographic Software of a public algorithm disclosed

66 If just a repetition of public sources without additional proprietary protective means took place, no IP-asset is included. Here the piece of software is de facto no asset.

67 Although the algorithm is public, how the things are implemented and protected by countermeasures against side channel and perturbation attacks could remain as an IP-asset. We assume here further proprietary modification of the public sources.

68 If somebody licenses[4] protective software, this means that it includes already countermeasures for implementing a public algorithm, it is a critical case by case decision after reassessment:
If this software with countermeasures becomes widely known and is no longer under control of the IP-holder, as a result of the disclosure, then security relevant countermeasures have been disclosed. Formally this is then public knowledge and no points could be given for it, in an attack potential calculation.

4.4.4 Case 8: Keeping a proprietary algorithm as an asset

69 Keeping a proprietary algorithm as a confidential asset is the trivial case: the software is clearly an IP-asset.

[4] A license in this sense means providing a third party the source code. The provision of object or executable code is different as this requires reverse engineering.

Overview Table

70 Following table summarizes the chapters of above and leads to the conclusion in the next chapter:

		Asset		Affect on
Case	**Object**	**Security**	**IP**	**Certificate**
	Firmware			
4.3.1	No secrets	No (1)	Optional	No
4.3.2	With secrets	Yes	Optional	Yes – recalculation at surveillance/reassessment
	Software			
4.3.3	No secrets	No (1)	Optional	No
4.3.4	With risk potential	Risk analysis	Optional	Yes – recalculation at surveillance/reassessment
4.3.5	With cryptographic secrets	Yes	Optional	Yes – recalculation at surveillance/reassessment
	Algorithm			
4.3.6	Without protective implementation	No	No	No
4.3.7	With protective implementation	Yes	Yes	Yes - surveillance/reassessment recalculation
4.3.8	Proprietary Algorithm	Yes	Yes	Yes – recalculation at surveillance/reassessment

(1) It must be considered on a case by case basis, if disclosed firmware without secrets would simplify or enable the mounting of a perturbation attack leading to a later exploit during the user software operation.

5 Conclusion

71 An observation of potential new attack path(s) based on public sources is recommended and also seen as an ongoing task for all scheme participating parties. For example, in Germany certificate holders are requested to proactively inform the BSI about emerging risks from the market, internet sources or other publications. All participants of the scheme monitor the public sources to certain extent.

72 Assumed that one party, either Certification Body, ITSEF, hardware or software developer, has gained awareness about a specific publication of disclosure of code related to a certified product, the rules for first action as proposed in chapter 4.1 apply.

73 The above scenarios include cases where firmware/software has been disclosed without endangering a given evaluation result or increasing the level of vulnerability. This is one of the major benefits of the rules proposed: they localize the threat of a third party disclosure to the relevant code parts and limits the affected certificates, products and projects. They separate IP concerns from certification concerns and thus separate the economic impacts of a disclosure on these different aspects.

74 In case a disclosed code part is not a security asset, but represents an IP-asset, it is rather a commercial decision whether a chip provides sufficient IP protection. An evaluation result or the security is not affected, as long as the IP-asset has not been defined as also being a security asset.

75 Following this case by case approach, the developer documents must contain clear definition or assignment about which parts of the software/firmware are security assets as opposed to IP-assets, so that only disclosure of the security assets will affect certification. The developer documents – firmware and software - must also identify which parts are reused and where those reused parts are taken from. This implies the declaration if reused code parts are also present on unprotected chips or originate from public sources.

76 This assignment of code parts need not be included in the public Security Target or other documentation available to users, as this could point an attacker towards interesting attack or research targets. Therefore, such declaration shall be part of the confidential developer documentation, e.g. it could be part of the TOE design description.

6 References and Abbreviations

[1] Security IC Platform Protection Profile, Version 1.0, 13.01.2014, BSI-CC-PP-0084-2014.

AP: Application (user software)
CB: Certification Body
FW: Firmware
ETR: Evaluation Testing Report
ICC: ICC key in hardware used for encryption/decryption of application secrets (PIN)
IP: Intellectual Property
ITSEF: Information Technology Security Evaluation Facility
OS: Operating System (user software)
SW: Software