Specification and Refinement of Secure IT-Systems — Extended Abstract —

Thomas Santen
Institut für Softwaretechnik und Theoretische Informatik
Technische Universität Berlin
10587 Berlin, Germany
santen@acm.org

Andreas Pfitzmann
Fakultät für Informatik
Technische Universität Dresden
01062 Dresden, Germany
pfitza@inf.tu-dresden.de

Maritta Heisel
Institut für Praktische Informatik und Medieninformatik
Technische Universität Ilmenau
Max-Planck-Ring 14, 98693 Ilmenau, Germany
maritta.heisel@prakinf.tu-ilmenau.de

1 Some Remarks on Formal Methods from a Security Perspective

Dependable IT-systems [11] of relevant size can only be built if all dependability attributes have a clear meaning. This meaning must be consistent both with the common understanding of the people building and using the IT-system, and with the tools they use and the development process they adhere to. Therefore, a formal meaning of all dependability attributes is needed which is compatible with the usual refinement process of systems engineering.

Consider the meaning of security attributes. Some relationships with established formal notions are clear whereas others are not understood so well. Confidentiality, integrity, and availability are seen as the generic properties of security [17, 5]. Some authors propose to add accountability as a fourth one [4], others propose to refine confidentiality, integrity and availability according to the kind of information they relate to [18]. Then, e.g. accountability can be interpreted as an integrity property concerning the circumstances of an action, anonymity can be interpreted as a confidentiality property concerning the circumstances of a communication, and so on.

The relationship between the security attributes integrity and availability, and formal notions compatible with refinement is roughly clear. For terminating computations, integrity corresponds to partial correctness, and availability corresponds to assured termination combined with sufficient computational resources to fulfill real-time requirements. Integrity and availability together correspond to total correctness and sufficient computational resources. For reactive systems, integrity means that the defined processes satisfy certain required predicates, and availability corresponds to fairness and liveness combined with sufficient computational resources.

To date, the relationship between confidentiality and formal notions of refinement is less well understood. One of the reasons is that a possibilistic framework is not enough to prove security in general and confidentiality in particular. Such a framework only makes it possible to find some security flaws. To positively prove security, a *probabilistic* framework is needed.

An Example: Protocol Verification Consider the state of the art of (cryptographic) "protocol verification". Approaches such as specially designed logics to describe and analyse protocols [3], and approaches that use proving techniques such as model checking [12] or interactive theorem proving [14] all have several deficiencies in common.

First, their verdict is based on a possibilistic analysis only. They may find flaws in protocols that definitely compromise a secret to an adversary. They will usually *not* complain about a protocol that leaks sufficient information for an adversary to deduce a secret with a probability of 99.9%. From a practical point of view, however, the latter protocol is as insecure as the former.

Second, the notion of security against which a protocol is analyzed usually remains implicit in the definition of the analysis. Therefore, it is unclear what positive reassurance a protocol "verification" that does not find a flaw in a given protocol actually establishes. One of the reasons for this kind of vagueness is that a possibilistic framework does not suffice to characterize security properties. Another reason is that security requirements – and those related to confidentiality in particular – often involve *implicitly negative* arguments: “the system is allowed to do only what is explicitly required – and *nothing else*”.

Security properties which are not achieved perfectly may be even harder to prove than perfect ones, but these non-perfect security properties are often the best you can achieve in practice. Then, counting arguments of equally probable possibilities are not enough.

Very large and varying numbers of participants may participate in future protocols. In addition, the security property to be proved may depend on the number of participants and even the development of their number. Nearly all security protocols providing anonymity properties in general and anonymous communication in particular are of that nature.

Future protocol runs may be long lived in the sense that in the midst of the protocol run, some participants may get dishonest (e.g. their Windows ME machine gets subverted), but others get honest (e.g. they re-install their Windows operating system and application software and disable ActiveX).

It is not enough to specify all security properties a protocol is intended to achieve, but in addition the complete environment, in particular all security relevant assumptions, have to be specified. If this is not done, you neither can hope that security can be proved nor that security protocols are compositional. An example what happens if relevant properties of the environment are ignored is a ”secure” remote login protocol (SSH), which transmits each keystroke of the human user in a separate message immediately: Even when all crypto is perfect (probabilistic encryption), the mere timing of the messages sent gives an attacker information he can exploit – the time intervals between messages correspond to the time intervals between keystrokes – and the time intervals give, when observed long enough and analyzed cleverly, all characters typed with significant probability. That way, an attacker gets valuable information on all passwords and on all pass phrases, e.g. to unlock cryptographic keys.

2 Ingredients for Security-Aware Software Engineering

While the formalisms and tools of the formal methods community are reasonably well suited for developing safety-critical systems, much is lacking to really support the development of secure IT-systems.

2.1 Specification

We need languages to express the specific requirements concerned with security. In our established formal languages (model-based or algebraic specification languages, process calculi, statecharts, etc.) – and also in not-so-formal modeling notations such as UML – one will search in vain for means to express *protection goals* and to model the capabilities of possible *adversaries*. A clear understandig of both is the key to assessing the appropriateness of security mechanisms with respect to the requirements for a given IT-system.

It is also not clear what a suitable semantic domain for such languages should be. As we have argued in the preceeding section, probabilistic models are necessary to really capture the meaning of confidentiality properties. If we take into account that some aspects of integrity, e.g. mutually establishing the authenticity of principals in protocols for e-commerce, are implemented using cryptographic tools such as authentication systems, it is clear that probabilistic arguments are also necessary for other aspects of security than confidentiality.

2.2 Refinement

Of course, just formally specifying the security requirements of an IT-system is not enough. For critical requirements, a rigorous notion of what it means that an implementation faithfully realizes those requirements is needed. Then, we can develop methods and tools to help demonstrate that an implemented system actually fulfils the security requirements, expressed as protection goals, against an adversary described by a suitable adversary model.

3 State of Formal Methods Research in Security

Much of current research of the formal methods community on security related topics addresses protocol verification [3, 12, 14]. Although, as we have argued above, “protocol falsification” might be a more suitable name for that branch of research, it does provide some insight in designs of secure protocols because tool-supported analyses using model checking or theorem proving can find non-obvious flaws that human inspection may not be able to find.

Another branch of research considers the implementation of security mechanisms such as authentication protocols, e.g. for the CORBA authentication service in the context of component based software engineering [2]. This work addresses the internal consistency of security mechanisms (“Is the service specification contradictory?”), and it can support demonstrating that an implementation is a functionally correct refinement of a service specification. It can-

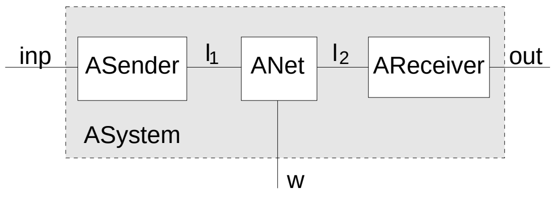

Figure 1. System Model

not, however, demonstrate that an implementation does not contain a “trojan horse” that allows an adversary to circumvent the protection that the service is design to establish.

Little research focusses on notions of “secure refinement”. Roscoe et. al. [15] have shown how non-interference [1, 6, 7] can be preserved by refinement for deterministic systems.

Mantel [13] considers the preservation of information flow properties under refinement. It is well-known that CSP-style refinement does not preserve information flow properties in general [9]. Mantel shows how refinement operators tailored for specific information flow properties can modify an intended refinement such that the resulting refinement preserves the given flow property. Working top-down from the specification to an implementation, the refinement operators may lead to concrete specifications that are practically hard to implement, because the changes in the refinement they induce are hard to predict and may not be easy to realize in an implementation.

Jürjens [10] uses stream processing functions to model systems. He defines a notion of secrecy which is only necessary in the sense that it does not prevent implicit information flows. His condition is also only necessary because an observing adversary gains a secret either completely or not at all. Jürjens does not consider that an adversary may gain information about the secret in a probabilistic sense. He identifies conditions under which certain refinement operators on stream processing functions preserve his notion of secrecy.

4 Confidentiality-Preserving Refinement – A First Step to Probabilistic Secure Refinements

As a first step towards a formally based refinement technique for secure systems that takes probabilities into account, we have developed a definition of *confidentiality-preserving refinement* [8]. To specify confidentiality properties we use a system model illustrated in Figure 1 for the case of a communication system between a sender and a receiver communicating over an untrusted network. We specify a system for which confidentiality is an important requirement by a pair of a process and a window channel. We use CSP extended with a probabilistic choice operator to specify processes.

Definition 1 (System, Window)
A system specification $S = (Q, w)$ is a pair of a process definition Q and a distinguished channel $w \in \alpha Q$, called the window *of S.*

The window channel w models the information flow from the system to an adversary. Observing the channel w, the adversary gains information about the system. Any distinction the adversary can make about the internal state of the system based on the observations on w is information that the system does not keep confidential. Conversely, the system keeps confidential any aspect of its behavior that an adversary cannot distinguish by observing w. We formally capture that confidentiality property by defining equivalences over system traces.

Definition 2 (Indistinguishability)
Let $S = (Q, w)$ be a system specification. Let win be the function projecting traces in $traces(Q)$ to traces of $Q \setminus (\alpha Q - \{w\})$, i.e. to the sequences of events on the window w. Two traces $s, t \in traces(Q)$ are indistinguishable *iff the projections to w are equal:*

$$s \equiv t \Leftrightarrow win\, s = win\, t$$

In the transition from an abstract to a concrete system specification, the interpretation of a window changes: The window of an *abstract* system specifies what information is *allowed* to be visible to the outside world. The window of a *concrete* system specifies what information *cannot* be hidden from the environment.

Here, a purely logical argument is insufficient because it is not enough to ask whether a distinction in the concrete system *definitely* allows an observer to distinguish confidential data, but we must describe whether such a distinction provides *more* information[1] about the confidential data than the abstract window reveals. Therefore, we consider the respective probabilities of internal data that may cause a particular observable behavior on a window. Figure 2 illustrates our approach to formalizing that probabilistic argument:

Let r and s be two abstract traces that are indistinguishable with respect to the window. According to R, trace r can be represented by the concrete traces v and w, and trace s can be represented by the concrete traces x and y, where v and x as well as w and y are indistinguishable by observing the concrete window. For keeping r and s indistinguishable in the concrete system, we must require that the probability that r is represented by v be the same as the probability that

[1]Note that information is a probabilistic notion [16].

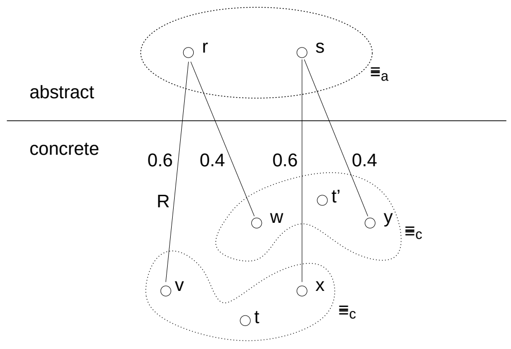

Figure 2. Probabilistic concretization and indistinguishability

s is represented by x. If this were not the case, an adversary might be able to gain information whether r or s happened on the abstract layer: if the probability that r is represented by v is greater than the probability that s is represented by x, for an adversary, the observation of some element $t \equiv_c v$ increases the probability of r with respect to s.

The following definition reflects this argument: A confidentiality-preserving refinement is one that (1) is a usual behavioral and data refinement of the processes describing a system, and (2) it (probabilistically) preserves the indistinguishability of system traces.

Definition 3 (Confidentiality-Preserving Refinement) *Let $A = (Q_a, w)$ and $C = (Q_c, w)$ be two system specifications. Let $\equiv_a$ be the observational equivalence in A (wrt. w), $\equiv_c$ be the observational equivalence in C (wrt. w), and let P_c be the probability distribution on $traces(Q_c)$. The system C is a* confidentiality-preserving refinement *of the system A iff there exists a retrieve relation R mapping the data of C to the data of A with inverse R^{-1} such that:*

1. *Q_c is a behavioral refinement of Q_a, i.e. $Q_a \setminus \{w\} \sqsubseteq_R Q_c \setminus \{w\}$, and*

2. $\forall\, r, s : traces(Q_a);\ t : traces(Q_c) \bullet r \equiv_a s$
$\Rightarrow P_{R^{-1}(r)}(u \equiv_c t) = P_{R^{-1}(s)}(v \equiv_c t)$

In Definition 3, the probability $P_{R^{-1}(r)}(u \equiv_c t)$ is the probability of choosing $u \in R^{-1}(r)$ such that u is indistinguishable from the given trace t of Q_c. For different traces r and s of Q_a, the sets of possible implementations $R^{-1}(r)$ and $R^{-1}(s)$ of r and s, respectively, may be different. Therefore, the probability distributions $P_{R^{-1}(r)}$ and $P_{R^{-1}(s)}$ can be different, too.

Properties of Confidentiality-Preserving Refinement

Confidentiality-preserving refinement is transitive [8]. Unfortunately, in general, it is *not* compositional, i.e. if C is a confidentiality-preserving refinement of A and we use A in a context $K(A)$, then it is in general not true that $K(C)$ is a confidentiality-preserving refinement of $K(A)$. The example in Figure 3 illustrates that observation.

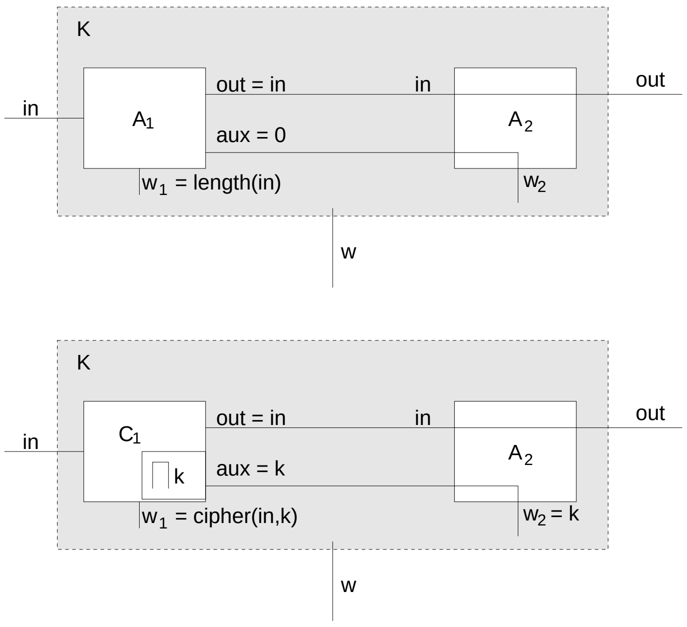

Figure 3. Confidentiality-preserving refinement is not compositional

In the abstract system, the component A_1 specifies a secure communication service. It lets its environment observe the length of messages transmitted from channel *in* to channel *out*, but no other information about the content of messages. The implementation C_1 is a confidentiality-preserving refinement of A_1. It randomly chooses keys k with equal probabilities to conceal the messages transmitted over the network. An adversary can observe the cipertext $cipher(in, k)$, which will reveal the lenght of the message but nothing else about its content.

In the context shown in Figure 3, replacing A_1 by C_1 does not preserve confidentiality, because the channel *aux*, which is of no use in A_1, is data-refined by C_1 to transmit the keys k to the second subsystem. That system makes the key visible to the environment over its window channel w_2 and thus compromises the confidentiality requirement of A_1.

5 Conclusions

We have argued that probabilistic definitions are necessary to capture security aspects (in particular confidentiality) in formal notions of refinement. We have proposed a definition of confidentiality-preserving refinement, which we consider the starting point of more research into secure refinement. It remains to find sufficient (or better: necessary and sufficient) conditions for such a refinement to be compositional. It is also clear that our definition is too strong for many cryptographic systems used in practice, because those systems are (inevitably) insecure in exponentially few cases. We are working on a variation of our notion of refinement that will accept those systems without admitting "truely" insecure refinements.

References

[1] J. Allen. A comparison of non-interference and non-deducibility using CSP. In *Proceedings of the 1991 IEEE Computer Security Workshop*, pages 43–54. IEEE Computer Society Press, 1991.

[2] D. Basin, F. Rittinger, and L. Viganò. A formal analysis of the CORBA security service. In *2nd International Z and B Conference (ZB 2002)*. Springer-Verlag, 2002. to appear.

[3] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.

[4] Canadian System Security Centre. The Canadian trusted computer product evaluation criteria (version 3.0e). Communications Security Establishment; Government of Canada, 1993.

[5] European Communities – Commission. *ITSEC: Information Technology Security Evaluation Criteria (Provisional Harmonised Criteria, Version 1.2, 28 June 1991)*. Office for Official Publications of the European Communities, Luxembourg, 1991. ISBN 92-826-3004-8.

[6] J. Goguen and J. Meseguer. Security policies and security models. In *Proceedings of the 1982 IEEE Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society Press, 1982.

[7] J. Graham-Cumming. Laws of non-interference in CSP. *Journal of Computer Security*, 2:37–52, 1993.

[8] M. Heisel, A. Pfitzmann, and T. Santen. Confidentiality-perserving refinement. In *14th IEEE Computer Security Foundations Workshop*, pages 295–305. IEEE Computer Society Press, 2001.

[9] J. Jacob. On the derivation of secure components. In *IEEE Symposium on Security and Privacy*, pages 242–247. IEEE Press, 1989.

[10] J. Jürjens. Secrecy-preserving refinement. In J. N. Oliveira and P. Zave, editors, *FME 2001: Formal Methods for Increasing Software Productivity*, LNCS 2021, pages 135–152. Springer-Verlag, 2001.

[11] J.-C. Laprie, editor. *Dependability: Basic Concepts and Terminology in English, French, German, Italian and Japanese*. Springer-Verlag, 1992.

[12] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, LNCS 1055. Springer-Verlag, 1996.

[13] H. Mantel. Preserving information flow properties under refinement. In *IEEE Symposium on Security and Privacy*, pages 78–91. IEEE Computer Society Press, 2001.

[14] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, pages 85–128, 1998.

[15] A. W. Roscoe, J. C. P. Woodcock, and L. Wulf. Non-interference through determinism. In D. Gollmann, editor, *European Symposium on Research in Computer Security (ESORICS)*, LNCS 875, pages 33–53. Springer-Verlag, 1994.

[16] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[17] V. L. Voydock and S. T. Kent. Security mechanisms in high-level network protocols. *ACM Computing Surveys*, 15(2):135–171, 1983.

[18] G. Wolf and A. Pfitzmann. Properties of protection goals and their integration into a user interface. *Computer Networks*, 32:685–699, 2000.