The unreasonable ineffectiveness of security engineering: An overview

Dusko Pavlovic
Kestrel Institute and Oxford University
Email: dusko@{kestrel.edu,comlab.ox.ac.uk}

***Abstract*—In his 1960 essay, Eugene Wigner raised the question of "the unreasonable effectiveness of mathematics in natural sciences" [32]. After several decades of security research, we are tempted to ask the opposite question: Are we not unreasonably ineffective? Why are we not more secure from all the security technologies? I sketch a conceptual landscape of security that may provide some answers, on the background of ever increasing dynamics and pervasiveness of software and computation.**

***Keywords*-security; formal methods; pervasive computation; networks**

I. Problem: All protocols are insecure

A. Failures are first class citizens

Security is often viewed as one among many (types of) requirements that may be imposed on a software system. Of course, the notion of security does not apply only to software, as we also speak of bank security, national security, secure locks etc. But since software nowadays permeates most domains that need to be secured, it is useful to think of security engineering as a part of software engineering. Within the framework of software architectures, security engineering can be viewed as the counterpart of the component based approach, since security protocols can be construed as abstract specifications of software connectors.

In practice, however, security engineering does not merge smoothly with the other engineering disciplines. The main reason is that in all other engineering disciplines failures usually arise from engineering errors, whereas in security engineering, they often arise *in spite of* the compliance with the best practices. Failures are not exceptions, but the first class citizens of security engineering! They are the heart of the problem of security.

B. Verified protocols fail

Formal verification is often viewed as the process of proving eternal mathematical truths about some given hardware or software artifacts. If I prove that a system is safe, then it will stay safe forever, unless I made a mistake, or unless my assumptions about the environment become invalid. Similarly, if I prove that a system is secure, it should remain secure forever.

Driven by the successes of formal verification of hardware and software, the formal methods community quickly adapted many of its powerful tools for reasoning about security [10], [25], [30]. But the practice of using these tools uncovered a remarkable phenomenon: many formally correct security proofs abstract away real attacks. In fact, almost every security proof eventually encounters a exception, or an attack, on one level or another.

Examples abound. The best known (although not the most informative) case is the Needham-Schroeder Public Key (NSPK) protocol [21]. It was proven secure in one formalism (BAN logic [5]), it was believed to be secure for 17 years, and then it was attacked in another formalism (CSP [17]). Thousands of papers cite NSPK as the crown evidence that formal reasoning is useful for uncovering attacks. But the fact is that different formal reasoning also concealed the attack on NSPK. In fact, BAN logic analysis did not conceal the attack, but eliminated it by assuming that the participants of the protocol are honest. If they are, the security proof is valid. The attack is launched by a dishonest responder. The NSPK story only shows that security is not an absolute property: the same protocol can be secure in one sense, and insecure in another sense. This is logically unsurprising, but it can be quite confusing in practice.

There are much deeper problems, though. Engineers generally build concrete complex systems incrementally, by refining and composing abstract and simple subsystems. But security is not preserved either under refinement, or under composition. This is well-known in theory, but easily forgotten in practice. A security proof may be valid for an abstract protocol, but invalid for some of its concrete implementations. Bull's recursive authentication protocol [4] was proven secure with a generic encryption operation [24], but found to be vulnerable to an easy attack in a slightly refined model, with a more concrete encryption operation [29]. In other cases, a protocol secure on its own may become insecure in interaction with other protocols [9].

But the problems of security do not boil down to the difficulties of composition and refinement of secure protocols. Even with the advantage of hindsight, it is not always

Supported by ONR.

possible to point to a single aspect of the verification process which led to a flaw: security always finds a new way to fail. The failures of the most popular formal model, generally accepted in the cryptographic community, have drawn the criticism that the complexity of that particular model is to be blamed: that too many people write convoluted formal proofs which too few people read; and that complex formalisms can be just as error prone as informal expert scrutiny [12], [11], [13]. In fact, both are error prone, and formal modeling often uncovers the vulnerabilities missed by many devoted experts, whereas the experts often capture the vulnerabilities missed in formal modeling. And both can happen even with a carefully scrutinized, and widely deployed protocol, included in many standards, and even formally modeled, such as MQV [15], [14], [20].

Clearly, combining formal methods with expert scrutiny gives the best of both worlds, and more vulnerabilities are captured then by either approach alone. Such process has been adopted in some of the working groups of the Internet Engineering Task Force (IETF), where the representatives from the industry stakeholders and the researchers from academia collaborate to adopt the internet standards. But the failures remain first class citizens nevertheless. E.g., the Group Domain of Interpretation (GDoI) of the IPSec protocol suite was both formally verified and thoroughly scrutinized by the best experts in the field, who reported about it in seven Internet Drafts, released in as many years, before standardizing it in [3]. Shortly after the standardization, an attack on the basic functionality of a version of GDoI was found [18].

II. Background: Security is an adversarial process

A. What is security?

It is well known since [1] that software dependability properties can be decomposed into

- *safety*: bad things don't happen, and
- *liveness*: good things do happen.

A similar conceptual dichotomy extends to information security, where the properties of interest are spanned by

- *secrecy*: bad information flows don't happen, and
- *authenticity*: good information flows do happen.

However, while the safety and the liveness of a system are independent, and in fact orthogonal properties, secrecy and authenticity essentially depend on each other:

- every secret must be authenticated, and
- every authentication is based on a secret.

The complexity of the reasoning about security is due to a great extent to this circular interdependency of authentications and secret sharing.

In fact, almost everything about security is circular.

B. The life cycle of security

In cryptography, there is a slogan that *every secret has a lifetime*: if it is not refreshed, it will be guessed. In security engineering, it seems, *every protocol has a lifetime*: sooner or later, an attack against it will be found. The protocol will then be improved to avoid that attack: the theory of security on which the protocol was based will be refined. In response, the attacker will seek a situation not captured by that new theory, and launch an attack as a counter-model for it.

$$\begin{tikzcd} \text{Protocol} \arrow[d, bend left, "\text{counter-model}"] \\ \text{Attack} \arrow[u, bend left, "\text{theory}"] \end{tikzcd}$$

This is the ongoing adversarial process of security.

The general *assume-guarantee* view of engineering is that the System guarantees a functionality, provided that the Environment satisfies its assumptions. This is useful when the Environment is passive, and does not change too often. The characteristic of security engineering is that the Environment actively seeks to invalidate System's assumptions. While security researchers plead allegiance to the requirement that all security assumptions and mechanisms should be transparent (”no security by obscurity”), security analyses often inspire attacks, by explicating the security assumptions, suitable for the attacks. Even publishing the patches to cover vulnerabilities triggers attacks. As a logical process, security is a game of imperfect information.

III. Gap: Methods for pervasive software design

A. Pervasive programming as protocol design

As computers pervade all areas of social life and production, computation itself is becoming a social process, distributed across the networks of computers, people, devices, and other networks. In contrast with the purpose-built electronic computers, these spontaneously evolving social computers cannot be directly controlled, or programmed in the traditional sense. Their global behaviors can only be steered indirectly, by constraining the local computational interactions. The program particles that assure that a system of local interactions leads towards the specified global goals can be viewed as security protocols. In this sense, security can be defined as a family of methods to achieve some global, social goals by means of some local, individual constraints. Security analyses thus tackle the complexity of global effects of local processes. This is why security becomes the central concern in network computation.

B. *The need for high level methods*

Computer science is the science of programming. Its main feat were high level programming languages, for expressing and translating human commands into machine code. Research into program abstraction led to the semantical revolution of the 1970s and 80s, which consolidated the mathematical underpinnings of computer programming into a foundation for a science. Software engineering organized programming methodologies into broader software development and management processes. The effectiveness of software engineering and programming methodologies is a consequence the robustness of their mathematical foundations.

But when computers got connected into networks, and networks into networks of networks, the science of programming did not follow. The high level methods for program design were not extended or adapted into methods for secure protocol design. The notions of program abstraction and specification refinement have not been adequately formalized for the modern forms of network computation. In this realm, the semantical revolution did not yet happen. The mathematical underpinnings of security research are still scattered through many unconnected research fields, and across several low level models. E.g. in cryptography, completely formalized proofs still are still often written in the low level language of Turing Machines. The high level descriptions are usually written in English.

C. *Pervasive computation*

The idea of pervasive computation can be traced all the way back to Vannevar Bush's Memex memo [6], which clearly anticipated the Web from a distance of 50 years. The visions of "mechanically extended man" [22] and "man-computer symbiosis" [16] were passionately discussed in the 1950s, before the idea of artificial intelligence was formulated. Building upon such visions, Doug Engelbart pursued in the 1960s a focused research of *"smart spaces"*, extending by computers human collaboration and collective intelligence [8]. Engelbart carefully analyzed the spatial aspect of pervasive computation, but his ideas were too far ahead of his time, and their practical realizations were even further. For a long time, the only practically useful result of his investigations was the mouse interface. Although it was not taken too seriously either, it caught on when the computer screens were structured as 2-dimensional spaces, and subdivided into windows, rather than into 80-character command lines, as they were before. Until recently, the screen was the only computationally relevant space: local computation was a stream of symbols, and the networks were spanned through a *distance-free* cyberspace. Engelbart's work was largely forgotten, except in the futuristic MIT "Media room" project from 1970es, which experimented with computational environments with implicit, sensor-based computational inputs.

The spirit of Engelbart's ideas was revived in the 1990s by Mark Weiser [31], under the name of *ubiquitous computation*. This time the emphasis was not on the spatial aspects of network computation (viewed as a social process, along the lines of Vannevar Bush's ideas), but on hiding the computer interfaces in everyday objects, so that computation disappears into the environment, and becomes an implicit activity.

The reality of the wireless networking technologies upstaged the predictions of all theories of pervasive computation, just like the reality of the Internet had upstaged the theories of distributed computation. Nowadays, networks of small, versatile computational devices overlay and amplify social networks. They promulgate through physical space, and communicate through their humans on their social channels, just like the humans communicate through their devices on their electronic channels.

D. *Pervasive software*

But the advances in device design, network technology and deployment have not been met by the advances in program design, system composition and security. Pervasive hardware has been adopted, it permeates many aspects of modern life, and dominates the markets. Pervasive software design is largely ad hoc. Pervasive security hardly exists.

The methodologies for pervasive software design include Web and network programming, iPhone application design and development, mashup building, and some other approaches to building widely distributed software systems. Pervasive software developers share many tools and languages with the Web developers. Both require a wide gamut of programming and scripting techniques, sometimes include extensive protocol and interface design efforts. In both cases, social engineering is a central rather than a marginal concern. However, while the Web development methodologies are well supported and carefully analyzed by the industry, and on the enterprise level, pervasive software has been left to the individual developers. It is built on a large scale, it is supported by some powerful development tools, yet it has not attracted much attention of the research community. As a consequence, an important part of modern software — perhaps the most important part — has not yet been covered by formal methods or by rational engineering methodologies. The resulting lack of assurance and security imposes limits the critical applications of the available technologies.

IV. Towards a solution: Semantics of networks

A. *Three paradigms of computation*

In *simple computation*, as realized, e.g. by a Turing Machine, a computer takes a string as the input, and returns a string as the output, both explicitly structured, and both at explicitly specified interfaces. Computation can thus be construed as mechanized symbol processing, or calculation. It is off-line, in the sense that its only interactions with the environment are the input and output operations.

In *network computation*, as realized by a computer network designed according to the ”end-to-end” architectural principle, the comfortable explicitness of the data structuring and of the interfaces is lost: data may be differently structured at different network nodes. Moreover, data may be transformed not only through calculation within a node, but also through communication between nodes. Computation as local calculation is thus extended by non-local communication, and its effects. Its interfaces may be distributed between multiple network nodes, and even redistributed by a process. Explicitly specified data structures and interfaces are thus replaced by dynamically changing semantics and evolutionary network structures. The user interface of a network-as-a-computer requires *meta-computation*, which includes *search*, *data mining*, *latent semantics* extraction, and other methods to gather and analyze statistics of data and information flows. The network indexing methods that underlie this meta-computation, are rapidly evolving into a branch of software engineering on its own. The high-level operations of *web programming* are crystallizing on top of the spidering and indexing practices, and some view them as the first basic operations of a ”machine language” for harvesting data from networks [23], [7].

Pervasive computation further extends the paradigm of network computation in two directions.

- On one hand, the simple ”end-to-end” network architecture, where the communicating computers are connected by insecure links[1], and all security requirements are pushed to the ”ends” — is replaced by the *heterogenous network models*, as increasingly diverse computational devices get connected through increasingly diverse communication links.
- On the other hand, the abstract notion of cyber-space, where every two nodes are each other’s neighbors as soon as there is a network route between them, is replaced by richer notions of space, since some of the network links implement a notion of distance, and some are even directly embedded into social and physical spaces [19], [33], [27].

In summary, the notion of computation has evolved from calculation and string processing, through end-to-end communication in cyber space, as the distance-free space of costless communication, to the rich interactions in physical and social spaces, that constitute our social life. What was originally a mechanization of a simple part of our most abstract symbol processing capabilities has evolved into a practical support for our most concrete social interactions.

[1]”End-to-end” means that the principals are at the network nodes, and the attacker controls the network links.

Remark: The term *cyber-space* usually denotes the abstract space of end-to-end networks: the location and the physical distance of the end nodes are irrelevant; every two nodes appear to each other as neighbors, as long as a connection can be established. Introducing mobile devices as carriers of computation introduces the concept of distance among the computational interactions. We use the terms *”cyber-network”* and *”end-to-end network”* interchangeably.

B. Security protocols in pervasive computation

A protocol is a distributed computational process, given with a set of desired runs, or the properties that the desired runs should satisfy. To prove security of a protocol we usually demonstrate that only the desired runs are possible, or that the undesired runs can be detected through participants’ local observations.

Security protocols are thus formally modeled within a process calculus, or with partially ordered multisets (pomsets) [28]. In order to model security protocols run in pervasive networks, we extend the pomset process model, used for analyzing security protocols in cyber networks [26]. The main complication is that in this previous work, the network itself was abstracted away, i.e. left implicit. More precisely, the service of routing and relaying messages was taken for granted, and hidden from the observer. As unobservable, this service was assumed to be controlled by the adversary. An attack in an end-to-end network can only be detected through local observations at the ends, and without insight onto the message delivery processes in-between.

In a pervasive network, the message delivery services cannot be abstracted away. The assumption that every two nodes are network neighbors is not justified any more: e.g., some devices may be internet nodes, some may have access to a cellular network, and may communicate through their human carrier. Some may be connected directly, others may have to seek a relay. To express such situations, we must make the network explicit.

Moreover, the toolkit of security primitives and security tokens, available to establish secure communication, is essentially richer in pervasive networks.

C. Principals and security tokens

The principals are the computational agents that control one or more network nodes, where they can send and receive messages. A principal can only observe (and use in his reasoning) the events that happen at his own network nodes. Principal’s observations of other principals’ actions are modeled as messages received over *social channels*.

Security tokens are the data used by the principals to realize secure communication. Informally, security tokens are usually divided in three groups:

- something you know: digital keys, passwords, and other secrets
- something you have: physical keys and locks, smart cards, tamper-resistant devices, or
- something you are: biometric properties, e.g. fingerprints, or written signatures, assumed to be unforgeable

The difference between these three types of security tokens lies in the extent to which they can be shared with others:

- what you know can be copied and sent to others,
- what you have cannot be copied in general, but can be given away, whereas
- who you are cannot be copied, or given away.

The most common end-to-end security goals are usually realized entirely by means of cryptographic software, and the principals only use the various kinds of secrets. This means that *a principal can be identified with the list of secrets that she knows*. If Alice and Bob share all their secrets, then there is no way to distinguish them by the challenges that can be issued in a standard cyber network . For all practical purposes, they have the same network identity.

In pervasive networks, on the other hand, security is also supported by cryptographic hardware: besides the secrets, a principal is also supplied with some *security devices*. They are represented as some of the network nodes, given to the principals to control. A dishonest principal (or an honest certificate authority) can relinquish control of a security device, and give it to another principal.

To capture the third and the strongest kind of security tokens, and distinguish the principals by who they are, we need some *biometric devices*. They are represented as network nodes. Principals' biometric properties, on the other hand, are represented as some of the network nodes as well, available to respond to the challenges from the biometric devices. The only difference of a biometric property p from the other network nodes given to a principal A to control is that p always remains under A's control, and cannot be given away to another principal. We call the networks equipped with biometric devices and biometric properties — biometric networks.

D. Networks

In modeling security, principals can be identified with their security tokens, since security tokens are the material that security is built from. Summarizing the preceding section, we can say that

- in end-to-end networks (or cyber-networks), the only security tokens are the secrets, and the principals are reduced to *what they know*;
- in pervasive networks, the security tokens also include some security devices, and the principals are identified not just by what they know, but also by *what they have*;
- in biometric networks, the security tokens furthermore include some biometric properties, and the corresponding biometric devices, needed to test them; the principals are identified not just by what they know, or what they have, but now we take into account *who they are*.

A *communication network* consists of

- *network graph* $\mathcal{N} = (L \rightrightarrows N)$, where N is the set of nodes, and L the set of links, partitioned into $\mathcal{N}_{mn} = \langle \delta, \varrho \rangle^{-1}(m,n)$ for $m, n \in N$,
- *channel types* $\mathcal{C}$, and the type assignment $\theta : L \longrightarrow \mathcal{C}$,
- *set of principals (or agents)* $\mathcal{A}$, partially ordered by the subprincipal relation $\leqslant$,
- *control* $\copyright : \mathcal{A} \longrightarrow \wp N$, such that the first of the following conditions is satisfied, and often also the second one:

$$\begin{aligned} A \leqslant B &\implies \copyright A \subseteq \copyright B \\ A \nleqslant B \wedge A \ngeqslant B &\implies \copyright A \cap \copyright B = \emptyset \end{aligned}$$

Remark: In a cyber network, the end-to-end assumption, that all security is done at the "ends" and any route "in-between" is as good as any other route implies that the network service can be reduced to an assumption that there is a single link between every two nodes, i.e. $\mathcal{N}_{mn} = 1$ for all m and n. Moreover, $\mathcal{C} = 1$, i.e. all channels are of the same type, insecure. So the only nontrivial part of the structure is $\copyright : \mathcal{A} \longrightarrow \wp N$. But controlling one network node or controlling another one makes no difference, because a message can always be sent from everywhere to everywhere. So the only part of the above definition visible in the process model needed for cyber security is the poset $\mathcal{A}$.

1) Cyber networks: principals are what they know: The fact that the principals can be identified with the lists of secrets that they know is represented by an inclusion $\Gamma : \mathcal{A} \hookrightarrow \mathcal{T}^*$, which we call *environment*. However, since a principal may learn new secrets when a process is run (or during a protocol execution), her environment may grow: at each state σ, she may have a different environment $\Gamma_\sigma A$ such that for every transition $\sigma_1 \to \sigma_2$ holds $\Gamma_{\sigma_1} A \subseteq \Gamma_{\sigma_2} A$. During a protocol execution, different principals may thus become indistinguishable, if they learn each other's secrets, since $\Gamma A = \Gamma B \Rightarrow A = B$. This means that the set of principals $\mathcal{A}$ may also vary from state to state in the execution: there is a family $\mathcal{A}_\sigma$, with the surjections $\mathcal{A}_{\sigma_1} \twoheadrightarrow \mathcal{A}_{\sigma_2}$ for every transition $\sigma_1 \to \sigma_2$, induced by identifying the principals that become indistinguishable. In this case, the model thus boils down to the one used in Protocol Derivation Logic [18], [26].

Digression: Network identities: One often hears about network identities and identity theft. But what exactly is a network identity? Formal reasoning requires a formal definition. The terms *principal* and *entity* are often used to denote the carriers of a network identity.

Statically, a network identity is a set of secrets. Two principals who know the same keys cannot be distinguished by cryptographic challenges. And cryptographic challenges are the only authentication method in cyber networks.

Dynamically, however, different principals may become identical after the keys which distinguish them have been published. Or a principal A, which was independent of (incompatible with) B, may become B's subprincipal, if A's keys which B did not know are published. In general, two different principals may become indistinguishable even without any action on their own, entirely through the actions of others. E.g., suppose that

$$A = P \cup Q \quad B = P \cup R \quad C = Q \cup S \quad D = R \cup T$$

If C publishes Q and D publishes R, then A and B will become indistinguishable. A dynamical view thus shows principals as *variable sets of keys*, i.e. sets of keys changing through possible worlds.

2) Pervasive networks: principals are what they have: A *pervasive network* is obtained by distinguishing, within a cyber network as defined above, a set of mobile nodes (i.e. security devices) $\widetilde{N}$, from the fixed nodes $\overline{N}$, so that $N = \widetilde{N} + \overline{N}$.

Besides the send, receive, and match actions, the process calculus now has two new kinds of actions, which allow each principal to:

- move a mobile node under his control, and reconnect it elsewhere in the network;
- pass control of a mobile node to another principal.

This means that the network connections and controls of the mobile nodes can dynamically change during a process run.

3) Biometric networks: principals are what they are: A *biometric network* is obtained by distinguishing, among the nodes of a pervasive network as defined above, two more sets

- $B_r \subseteq \widetilde{N}$ of *biometric properties*, and
- $B_c \subseteq N$ of *biometric verifiers*.

The intended interpretation of these two sets of nodes is implemented by ther requirement that:

- control of the elements of B_r cannot be passed to another principal,
- the elements of B_c are related with the elements of B_r, so that the former can issue biometric challenges to the latter.

E. Message delivery modes

The main source of the new security phenomena in a pervasive network is the fact that the different types of channels have different message delivery modes. In cyber networks, a message is usually in the form A to $B : m$, where A is the claimed sender, B the purported receiver, and m the message payload. As explained before, the network service is implicit in this model, so that A and B refer both to the principals and to the network nodes that they control. All three message fields can be read, intercepted, and substituted by the attacker. The point of the end-to-end security is that the receiver can still extract some assurances, even from a spoofable message, because the various cryptographic forms of m limit attacker's capabilities. Moreover, this message form is an abstract presentation of the fact that the message delivery service provided by the network and the transportation layers, say of the Internet.

In pervasive networks, different channel types provide different message delivery services. In general, there is no universal name or address space, listing all nodes. Annotating all messages by sender's and receiver's identities thus makes no sense, and the principal's identities are added to the payload when that information is needed. There may be no link between two nodes, and no way to send a message from one to the other. On the other hand, a message can be delivered directly, e.g. when a smart card is inserted into a reader, without either of the principals controlling the card and the reader knowing each other.

The different message delivery modes determine the different security guarantees of the various channel types.

V. Conclusion

In networks and in pervasive computation, the desired global behaviors need to be assured by means of local constraints. The tasks of assuring global behaviors through local constraints subsume under security. This is why the central problems of pervasive computation usually revolve around security.

But security does not yield to the standard engineering techniques of incremental system development, based on refinement and composition. One reason, discussed in Sec. II, is that security is an intrinsically adversarial process: the Environment can actively change in order to hamper System's functioning. Another reason, discussed in Sec. III, is that the high level semantical methods for reasoning about pervasive computation and security are still largely missing. A sketch of a network model around which the needed semantical methods could be developed is provided in Sec. IV.

References

[1] B. Alpern and F. B. Schneider, “Defining liveness,” *Information Processing Letters*, vol. 21, no. 4, pp. 181–185, October 1985. [Online]. Available: http://dx.doi.org/10.1016/0020-0190(85)90056-0

[2] R. Barua and T. Lange, Eds., *Progress in Cryptology - INDOCRYPT 2006, 7th International Conference on Cryptology in India, Kolkata, India, December 11-13, 2006, Proceedings*, ser. Lecture Notes in Computer Science, vol. 4329. Springer, 2006.

[3] M. Baugher, B. Weis, T. Hardjono, and H. Harney, “The Group Domain of Interpretation,” Network Working Group, Internet Engineering Task Force. RFC 3547, July 2003.

[4] J. Bull, “The authentication protocol,” APM Report, March 1997.

[5] M. Burrows, M. Abadi, and R. Needham, “A logic of authentication,” *ACM Transactions on Computer Systems*, vol. 8, no. 1, pp. 18–36, 1990.

[6] V. Bush, “As we may think,” *Atlantic Monthly*, vol. 176, no. 1, pp. 101–108, July 1945.

[7] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[8] D. Engelbart, “Augmenting human intellect: A conceptual framework,” http://sloan.stanford.edu/MouseSite/Engelbart-Papers/B5_F18_ConceptFrameworkInd.html, October 1962.

[9] J. Kelsey, B. Schneier, and D. Wagner, “Protocol interactions and the chosen protocol attack,” in *Proceedings of the 5th International Workshop on Security Protocols*, ser. Lecture Notes in Computer Science, vol. 1361. Springer-Verlag, 1997, pp. 91–104.

[10] R. A. Kemmerer, C. Meadows, and J. K. Millen, “Three system for cryptographic protocol analysis,” *J. Cryptology*, vol. 7, no. 2, pp. 79–130, 1994.

[11] N. Koblitz and A. Menezes, “Another look at ”Provable Security”. II,” in *INDOCRYPT*, ser. Lecture Notes in Computer Science, R. Barua and T. Lange, Eds., vol. 4329. Springer, 2006, pp. 148–175.

[12] ——, “Another look at ”Provable Security”,” *J. Cryptology*, vol. 20, no. 1, pp. 3–37, 2007.

[13] ——, “The brave new world of bodacious assumptions in cryptography,” *Notices of the American Mathematical Society*, vol. 57, no. 3, pp. 357–365, Mar. 2010. [Online]. Available: http://www.ams.org/notices/201003/

[14] H. Krawczyk, “HMQV: A high-performance secure Diffie-Hellman protocol,” in *Protocol, Advances in Cryptology (CRYPTO) ’05*, ser. Lecture Notes in Computer Science, vol. 3621. Springer-Verlag, 2005, pp. 546–566.

[15] L. Law, A. Menezes, M. Qu, J. A. Solinas, and S. A. Vanstone, “An efficient protocol for authenticated key agreement,” *Des. Codes Cryptography*, vol. 28, no. 2, pp. 119–134, 2003.

[16] J. Licklider, “Man-computer symbiosis,” *IRE Transactions on Human Factors in Electronics*, vol. 1, pp. 4–11, Mar. 1960.

[17] G. Lowe, “An attack on the Needham-Schroeder Public-Key authentication protocol,” *Information Processing Letters*, vol. 56, pp. 131–133, 1995.

[18] C. Meadows and D. Pavlovic, “Deriving, attacking and defending the GDOI protocol,” in *Proceedings of ESORICS 2004*, ser. Lecture Notes in Computer Science, P. Ryan, P. Samarati, D. Gollmann, and R. Molva, Eds., vol. 3193. Springer Verlag, 2004, pp. 53–72.

[19] C. Meadows, R. Poovendran, D. Pavlovic, L. Chang, and P. Syverson, “Distance bounding protocols: authentication logic analysis and collusion attacks,” in *Secure Localization and Time Synchronization in Wireless Ad Hoc and Sensor Networks*, R. Poovendran, C. Wang, and S. Roy, Eds. Springer Verlag, 2006.

[20] A. Menezes and B. Ustaoglu, “On the importance of public-key validation in the mqv and hmqv key agreement protocols,” in *INDOCRYPT*, ser. Lecture Notes in Computer Science, R. Barua and T. Lange, Eds., vol. 4329. Springer, 2006, pp. 133–147.

[21] R. M. Needham and M. D. Schroeder, “Using encryption for authentication in large networks of computers,” *Commun. ACM*, vol. 21, no. 12, pp. 993–999, 1978.

[22] J. North, “The rational behaviour of mechanically extended man,” Wolverhampton, UK, Sep. 1954.

[23] D. A. Patterson, “Technical perspective: the data center is the computer,” *Commun. ACM*, vol. 51, no. 1, p. 105, 2008.

[24] L. C. Paulson, “Mechanized proofs for a recursive authentication protocol,” in *Proceedings of CSFW ’97*. IEEE Computer Society, 1997, pp. 84–95.

[25] ——, “Proving properties of security protocols by induction,” in *Proceedings of CSFW ’97*. IEEE Computer Society, 1997, pp. 70–83.

[26] D. Pavlovic and C. Meadows, “Deriving secrecy properties in key establishment protocols,” in *Proceedings of ESORICS 2006*, ser. Lecture Notes in Computer Science, D. Gollmann and A. Sabelfeld, Eds., vol. 4189. Springer Verlag, 2006.

[27] ——, “Deriving authentication for pervasive security,” in *Proceedings of the ISTPS 2008*, J. McLean, Ed. ACM, 2008, 15 pp.

[28] V. R. Pratt, “The pomset model of parallel processes: Unifying the temporal and the spatial,” in *Seminar on Concurrency, Carnegie-Mellon University*. London, UK: Springer-Verlag, 1985, pp. 180–196.

[29] P. Ryan and S. Schneider, “An attack on a recursive authentication protocol. a cautionary tale,” *Inf. Process. Lett.*, vol. 65, no. 1, pp. 7–10, 1998.

[30] ——, *The modelling and analysis of security protocols: the CSP approach*. Addison-Wesley Professional, 2000.

[31] M. Weiser, “Some computer science issues in ubiquitous computing,” *Commun. ACM*, vol. 36, no. 7, pp. 74–84, 1993.

[32] E. P. Wigner, “The unreasonable effectiveness of mathematics in the natural sciences,” *Communications on Pure and Applied Mathematics*, vol. 13, no. 1, pp. 1–14, February 1960.

[33] F. L. Wong and F. Stajano, “Multichannel security protocols,” *IEEE Pervasive Computing*, vol. 6, no. 4, pp. 31–39, 2007.