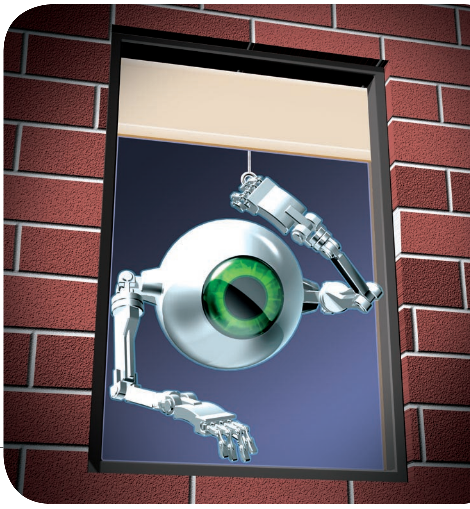
Challenges for Trusted Computing

Trusted computing is proving to be one of the most controversial technologies in recent years. Rather than become embroiled in the debate over possible (mis)appropriations of its technologies, the authors highlight some of the technical obstacles that might hinder trusted computing's widespread adoption.



Shane Balfe, Eimear Gallery, Chris J. Mitchell, and Kenneth G. Paterson
Royal Holloway, University of London

Trusted computing—the collection of interrelated and interoperating technologies that, when combined, help establish a more secure operating environment on commodity platforms—is undoubtedly a powerful technology with a huge range of possible applications. (In this article, a *trusted platform* refers to the type of platform championed by the Trusted Computing Group [TCG], "one which will behave in a particular manner for a specific purpose" [www.trustedcomputinggroup.org].) A fully realized trusted computing platform will let users reason about a platform's behavior and provide standardized mechanisms to protect sensitive data against software attack. Based on these capabilities, trusted computing has been proposed as a means of enhancing application security. For example, it has been promoted as an adjunct to the digital signature process, to enable secure software download, support secure single sign-on solutions, secure peer-to-peer networks, improve the security and privacy of biometric user authentication, harden mobile devices, and facilitate identity management. Several researchers have also considered trusted computing's applicability to the agent paradigm, grid security, e-commerce transaction security, and crimeware defense.

Despite its many potential beneficial applications, trusted computing isn't without detractors, especially relating to privacy concerns. The extent to which trusted computing could enable and enforce digital rights management and, more generally, the possible expropriation of platform owner control, are contentious issues. Critics have expressed concerns that trusted computing could support censorship, stifle competition between software vendors, facilitate software lock-in, and hinder open source software's deployment and use, thereby potentially enabling certain vendors to monopolize markets.

Our aim here isn't to engage in this debate, but rather to highlight some of the key challenges we believe must be tackled to accelerate trusted computing's widespread adoption. In particular, we address issues with setting up and maintaining the public-key infrastructure (PKI) required to support the full set of trusted computing functionality; the practical use and verification of attestation evidence; and backward compatibility, usability, and compliance issues.

Trusted Computing Technologies

Trusted computing relies on the successful integration and interoperation of several technologies:

- The TCG defines *a trusted platform module* (TPM) as a microcontroller with cryptographic coprocessor capabilities. A TPM provides several features: special-purpose registers called platform configuration registers (PCRs), which store information characterizing the host platform's configuration; a means of reporting the platform's current configuration to remote entities; secure volatile and nonvolatile memory; random number generation; a SHA-1 hashing engine; and asymmetric key-generation, encryption, and digital-signature capabilities.
- A *root of trust for measurement* (RTM) enables a host platform's configuration to be reliably recorded.
- *Isolation technologies*, such as Microsoft's Next Gen-

eration Secure Computing Base (NGSCB) or Citrix's XEN, take advantage of CPU and chipset extensions incorporated in new-generation processor hardware, including Intel's TXT and AMD's AMD-V. These technologies enable the unhindered execution of software through the provision of assured memory space separation between processes.

The ability to reason about platform behavior and protect sensitive data both rely on the concept of an *integrity measurement*, namely the cryptographic digest (or hash) of platform software. For example, a program's integrity measurement could be calculated by computing the cryptographic digest or hash of its instruction sequence (the executable), initial state, and input. However, in isolation, individual measurements of software components might be of little interest. To reason effectively about a particular platform component's behavior, the entire sequence of events that culminated in that component's execution must be measured. For example, during an authenticated boot process, initiated by the RTM, a platform's entire configuration is reliably captured and stored. During this process, the integrity of a predefined set of platform components is measured. These measurements are condensed to form a set of integrity metrics, which are then stored in the TPM's PCRs. These integrity metrics can be communicated to external entities for examination and verification via a process called *attestation*.

To protect sensitive data against software attack, the data can be associated with a set of integrity metrics representing a particular platform configuration or a password, and then encrypted. A TPM then ensures that protected data can only be decrypted and released if a user inputs the correct password or the platform's current configuration matches the integrity metrics sealed with the data. Deploying isolated execution environments further enhances sensitive data protection.

PKI and Trusted Computing

The development of any functional PKI requires a sophisticated combination of organizational, policy-oriented, procedural, and legislative approaches. Indeed, the challenges and pitfalls of PKI deployment are well-documented,[1,2] and high-profile system and protocol failures blamed on an inappropriate deployment of PKI abound, with the Secure Electronic Transaction (SET) protocol providing one of the most prominent examples. Put simply, providing a PKI is hard.

Most trusted computing services depend fundamentally on the deployment and successful interoperation of certain PKI elements. We refer to this collection of components as a trusted computing PKI (TC-PKI), although it's actually a larger and more complex ecosystem of elements than would normally be contained in a single PKI. Figure 1 depicts the main types of certification authority (CA) in a TC-PKI.

For a platform to be considered trusted, it must first obtain the following core credentials:

- *An endorsement credential*. Each TPM is associated with a unique asymmetric encryption-key pair called an *endorsement-key* (EK) pair. An endorsement credential binds this key pair's public component to a TPM description and vouches that a TPM is genuine. The endorsement CA is typically the TPM manufacturer, with the binding taking the form of a digital signature created using the manufacturer's signing key.
- *One or more conformance credentials*. Such credentials vouch that a particular type of TPM and associated components (such as an RTM and the connection of the RTM and TPM to a motherboard) conform to the TCG specifications. Conformance CAs must be entities with sufficient credibility to evaluate platforms containing TPMs and are typically conformance-testing facilities.

Trusted Computing Acronyms

- TCG: Trusted Computing Group
- PKI: Public-key infrastructure
- TPM: Trusted platform module
- PCR: Platform configuration register
- RTM: Root of trust for measurement
- TC-PKI: Trusted computing PKI
- CA: Certification authority
- EK: Endorsement key
- AIK: Attestation identity key
- P-CA: Privacy CA
- DAA: Direct anonymous attestation
- SKAE: Subject-key attestation evidence
- CP: Certificate policy
- CPS: Certification practice statement
- CRL: Certificate revocation list
- OCSP: Online Certificate Status Protocol

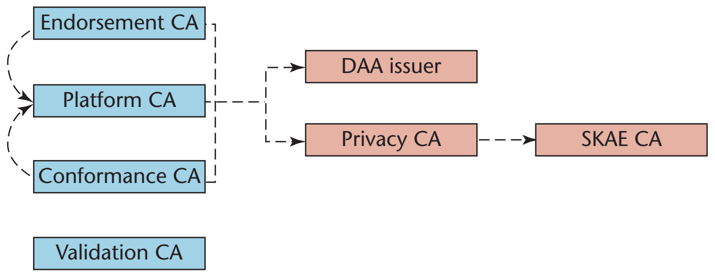


Figure 1. The trusted computing PKI.

- *A platform credential*. This credential asserts that a TPM has been correctly incorporated into a design conforming to the TCG specifications. The platform CA is typically the platform manufacturer. To create a platform credential, the platform CA must examine the endorsement credential and the conformance credentials of the platform to be certified.

Together, an endorsement, a platform, and one or more conformance CAs are responsible for issuing the core trusted-platform credentials. However, to address privacy concerns resulting from an EK's routine use, the TCG introduced the ability for a TPM to generate and use an arbitrary number of platform pseudonyms, in the form of *attestation identity key* (AIK) pairs. For a relying party to have assurance that an AIK represents a trusted platform, the platform must obtain an AIK certificate from a mutually trusted third party. The TCG has proposed two approaches to AIK certification.

In the first approach, a trusted third party, or *privacy CA* (P-CA), verifies a trusted platform's core credential set and provides assurance that an AIK is bound to a genuine trusted platform in the form of an AIK credential. However, this approach has attracted criticism because a P-CA can link all the AIK credentials it issues to a specific platform via the EK, putting the P-CA in a position where it might defeat the anonymity protection that AIKs provide.

The second approach, *direct anonymous attestation* (DAA), was introduced by the TCG to address this criticism. DAA requires a DAA CA, which can produce an anonymous DAA credential for a trusted platform, which the platform can in turn use to sign AIK credentials. Using this approach, trusted platforms can generate and use AIKs that a third party can't easily link to a particular EK.

Yet another class of CA has been introduced to attest to the usage, mobility, and authorization constraints associated with private keys that a TPM holds. A subject-key attestation evidence (SKAE) CA is responsible for issuing X.509 certificates that let a verifier ascertain that an operation involving a private key can only be performed within a TCG-compliant TPM environment. Such a certificate can help adopters cope with the difficulties of integrating TPM-controlled keys with standard security protocols. Recently, the TCG has introduced further PKI-related authorities (notably Migration Authorities and Migration Selection Authorities) to address issues with key migration between TPM-enabled platforms.

A TC-PKI not only involves a plurality of CAs but also a series of implicit dependencies among them. In a TC-PKI, a platform CA relies on an endorsement CA's due diligence and one or more conformance CAs to accredit a trusted platform's components. Similarly, both P-CAs and DAA CAs rely on platform CAs, endorsement CAs, and one or more conformance CAs. Furthermore, SKAE CAs rely on the due diligence of P-CAs or DAA CAs in evaluating the accreditation evidence that a trusted platform provides.

Traditionally, CAs deploy certificate policies (CPs), which specify what a certificate should be used for and the liability the CA assumes for this use, and certification practice statements (CPSs), which specify the practices that a CA employs to manage the certificates it issues, to define and limit their liabilities to relying parties. In fact, CPs and CPSs are an essential component in building a successful PKI because they give a relying party (such as a user or another CA) a means for managing the business risk in pursuing a particular PKI-related course of action. In the past, uncertainty as to where liability lies has driven up the cost of many PKI implementations.[2] In the absence of CPs and CPSs, implicit cross-certification might exist between CAs, which implies that CAs are equally trusted.[1] Such an approach reduces a certificate's security to that of the least trustworthy CA. Unfortunately, CPs and CPSs are notoriously difficult and costly to create, so their production might act as a barrier to entities wishing to provide CA services.

In a TC-PKI setting, every CA with dependent CAs must produce such policy statements. Currently, these dependencies are only informally defined, and consequently, it becomes unclear where any liability lies. Furthermore, at the time of writing, we're unaware of any TC-specific CPs or CPSs. The picture is additionally complicated because all the CAs in a TC-PKI rely (at least to some extent) on the endorsement CA. Therefore, the point in a TPM's life cycle at which an EK credential is acquired impacts a platform's ability to obtain platform, AIK, DAA, and SKAE credentials. In early normative EK credential acquisition, as defined by the TCG, a TPM manufacturer generates the EK credential. However, in postmanufacturing generation, a platform owner is responsible for generating the EK credential. In this instance, other CAs might not recognize the certifying body, and as a result, the certified TPM host platform might not be able to obtain further credentials from entities outside the domain of its EK credential issuer. In practice, this might not be an issue because it seems likely that nonmanufacturer-supplied EK credentials won't be widely used.

Therefore, trusted computing relies on an as yet largely unavailable and unspecified PKI in which multiple CAs (possibly existing in different organizational, procedural, and jurisdictional domains) are expected to interoperate. This poses a significant challenge to this technology's future success.

Credential and TPM Revocation

Revoking credentials within a TC-PKI might also introduce problems. Given the complex dependencies between many of the TC-PKI credentials, the compromise of an individual key and the subsequent revocation of its associated public-key certificate will result in a cascading revocation of all dependent TPM credentials. For example, in the event of EK revocation, every AIK associated with the revoked EK must also be revoked, in addition to all SKAE certificates associated with the newly revoked AIKs. This implies that multiple CAs, potentially in independent domains, must be contacted in a timely manner and informed about a revocation decision, which is likely to be a time-consuming and costly endeavor. Further complexity is introduced when attempting to revoke a DAA credential associated with a compromised EK pair because, by design, a DAA CA can't link a platform's EK pair with an AIK credential.

We next consider revocation of a TPM itself (rather than revocation of its credentials). For cost reasons, the level of tamper-resistance that TPMs provide is likely to be limited. Moreover, the objective of the TCG mechanisms is to prevent information asset compromise through software attack. That is, the platform's software security is predicated on the notion that the TPM will maintain an accurate and reliable record of all platform events. Yet, based on the evidence of widespread gaming-console modification, we know that users will actively circumvent hardware-enforced security given sufficient incentive. This poses a significant challenge to trusted computing, particularly given the existence of a relatively unsophisticated hardware attack that resets a TPM's PCRs without rebooting the platform.[3] Resetting PCRs effectively destroys the evidence that a remote verifier relies on to assess a platform. Once this evidence has been destroyed, the PCRs can in theory be repopulated with whatever data the platform owners wish, allowing them to misrepresent their platform's current configuration in a manner that's convincing to a remote verifier. Such a simple attack underlines the need for any verifier to consider the platform's "quality" when assessing its state.[3] That is, an attestation from a platform incorporating a well-designed TPM from a known manufacturer is more convincing than an attestation from a platform incorporating a TPM from an unknown or disreputable supplier.

Given this discussion, it's reasonable to assume that, before long, attackers will compromise TPMs and extract all credentials and keys. They could then use them to emulate a TPM in software in a way that's indistinguishable from the true hardware TPM. The process by which a compromised TPM is detected will largely rely on that TPM's interactions with P-CAs, DAA CAs, and SKAE CAs. Prior research has suggested that TPM compromise could manifest itself through an excessive number of certification requests originating from a single TPM host platform (where a risk-management policy determines what is "excessive").[4]

Unfortunately, this detection approach in turn introduces numerous challenges:

- CAs might specify different thresholds for determining what is excessive, potentially leading to a high number of false positives for CAs with low thresholds.
- Once a CA has detected a compromised TPM, this information must be globally propagated to prevent the compromised TPM host platform from being (mis)used elsewhere. This requires the establishment of a global revocation infrastructure that could be implemented using certificate revocation lists (CRLs) or through an Online Certificate Status Protocol (OCSP). Neither option, however, is ideal. With CRLs, there are concerns regarding CRL discovery and timely issuance of revocation information. To make OCSP deployment economically viable, CAs typically charge for each revocation check. It's unclear who would pay for such a service in a TC-PKI. In the case of an OCSP request for an SKAE certificate, the verifier would need to contact the SKAE CA, which would need to contact the AIK CA, which in turn would need to contact the platform, endorsement, and conformance CAs.
- A CA must consider potential legal issues that might result from wrongfully issuing revocation statements that damage a platform's ability to interact with other parts of the infrastructure. Consequently, CAs might be reluctant to announce suspected compromises.
- To alleviate the risk of a malicious CA issuing falsified revocation statements, we require a means of assessing the credibility of CAs issuing such statements. Currently, it's unclear what form such a mechanism might take.

Consequently, the development of a trusted computing PKI and the supporting key life-cycle policies and practice statements pose numerous legal and policy-design challenges, many of which remain largely unaddressed.

Attestation Evidence Gathering and Verification

Currently, the exact parameters to be considered when performing integrity measurements on platform components have yet to be standardized. At a minimum, the parameters to be measured must be chosen so that each software component's integrity measurement can be uniquely identified. These measurements must also remain consistent to allow ease of verification. However, in the absence of standardization, platform integrity measurements might fail to capture all the elements a platform component's verifier requires. This is especially true when we consider the complications introduced by software that relies on dynamically linked libraries. In this case, a proportion of the platform component's code base might not be measured because the required application code will not be loaded prior to execution.

Moreover, given the extensible nature of modern computing systems, the number of components that a TPM might need to measure is rapidly increasing. Consequently, each PCR will have to store multiple measurements. As the number of platform components increases, so does the complexity of third-party verification of attestation statements. It also becomes difficult for a challenger to verify a single component running on a platform.

Isolation technologies allow a platform to be partitioned into isolated execution environments, thereby (potentially) simplifying attestation-statement verification. In this case, the platform's challenger might be satisfied to verify measurements pertaining to rudimentary platform components, such as the boot software, the isolation layer, and software components running in an isolated execution environment, rather than verify all software running on the platform. This might ease the platform attestation problem in some situations.

However, even if we assume that the number of platform component integrity measurements that a challenger must verify is limited, problems will still arise from platform component updates and patching. Given current software development practices, we should expect frequent patching of OS components and applications to be the norm for the foreseeable future. But even the order in which patches are applied can result in a combinatorial explosion of distinct configurations for a single application, with each configuration requiring a distinct reference value for attestation purposes. Frequent patching might also lead to problems with sealed data. If an update or patch is applied to a software component with a sealed key or data, this key or data must be unsealed and resealed to the updated software component measurements. Failure to do so will make the key or data inaccessible after the patch has been applied.

Researchers have proposed *property-based attestation* to address the problem of managing attestation in the presence of many possible configurations and system updates.[5] This approach introduces an additional layer of indirection into the attestation and sealing processes. Instead of expecting a verifier to determine if a particular set of PCR values represent a trustworthy software state, a trusted third party certifies that a platform's state satisfies certain properties. A platform can then attest that its current configuration possesses such a property, letting a verifier infer whether a platform is trustworthy without knowing which particular software is running. Property-based attestation also allows a platform to seal data or keys to properties. This can reduce patching-related problems as long as the updated platform configurations' properties match those of the prior configuration.

Unfortunately, property-based attestation only succeeds in shifting the problems with attestation to an entity other than the verifier, with all the original problems persisting for the entity that needs to verify a PCR-based attestation. Moreover, a software component that satisfies a particular property is by no means guaranteed to still satisfy that property after it has been patched, without rerunning the (potentially expensive) evaluation procedure. Such an evaluation procedure could end up marginalizing minority platforms because the cost of establishing that a given platform state matches some desirable property might be so great that only a few well-funded organizations are able to obtain such a result. Also, exactly which properties can be attested using such an approach remains an open question.

More positively, property-based attestation at least shifts the problems to an expert specializing in attestation. This approach could significantly reduce the number of entities needing to verify such complex attestations, and these entities could receive additional resources to help them complete their task.

A final potential limitation of platform attestation is that of user-observable verification. McCune and his colleagues describe a scenario in which a user's platform has become infected with malware.[6] Even though an external entity can detect this infection during an attestation process, the external entity has no way of reliably informing users that they've failed their attestation. Malware might simply modify the user's display, resulting in the user believing that his or her platform is in an acceptable state and, because of this, disclosing sensitive information to the malware.

Backward Compatibility

As a consequence of the piecemeal rollout of trusted computing technologies, current trusted platforms don't come equipped with fully integrated RTMs, isolation technologies, processors, and chipset extensions. Instead, many current trusted platforms include only a TPM and, with the exception of Infineon TPMs, do not even include endorsement credentials. To the best of our knowledge, all currently available platforms lack both conformance and platform credentials. This situation could create an awkward backward-compatibility issue when fully deployed TC-PKIs become available. In particular, the absence of these credentials will make it difficult, if not impossible, for a platform to later acquire AIK credentials without operating at reduced assurance levels.

The absence of such protections from current TPM-enabled platforms makes the use of much of the TPM trusted computing functionality we've described essentially unreliable. Techniques such as sealing and attestation are unworkable if the host platform's configuration cannot be reliably measured. Enabling these features on an already deployed platform would mean integrating measurement functionality (in the form of an RTM and modified operating system) into the platform. This would require that we install a new operating system and flash the BIOS, which are difficult tasks for the average user.

On the other hand, this might be feasible in a corporate environment with centralized administrative platform control. Indeed, in such deployments, legacy hardware issues might be less serious because of more rapid platform upgrades or retirements. Moreover, an administrator could provide software-based isolation environments by installing additional software on already deployed platforms.

Nevertheless, we cannot retrofit hardware-based isolation, enabled through the processor and chipset extensions, to platforms already in the field. Consequently, first-generation trusted platforms can never be adequately upgraded to provide all the services associated with a trusted platform.

Usability

Prevailing wisdom suggests that it's prudent to hide the complexities of security technology from users. In the past, applications that relied on a PKI failed in cases where security functions were too unwieldy for nonexperts. In one example,[7] some users found the PKI experience so painful that they refused to use the technology if it involved handling certificates. The design of suitable user interfaces that can communicate rich security information while remaining usable has been historically difficult to achieve.[8]

By contrast, using a TPM currently requires a detailed understanding of how the underlying technology works. For example, the very act of enabling a TPM prior to its use is a nontrivial task requiring a user to understand and edit BIOS settings. Once enabled, a user is further confronted with setting a TPM owner password, selecting key types fit for purpose, and enrolling certain keys within a PKI. Further problems might arise from password use and management. In addition to setting a password for TPM ownership, unique passwords might also be associated with protected data or keys in a TPM. Although security experts might view deploying numerous passwords a sound security decision, user management of such passwords without jeopardizing access might prove problematic.

These usability issues are a reflection of the general immaturity of trusted computing technology and the associated marketplace. Although TCG has put much effort into the design and specification of trusted computing's technical aspects, there has been less work on addressing user-centric issues. We can hope for user-friendly configuration and management tools in the future, but even these might not be sufficient to make trusted computing accessible to the masses.

Addressing the obstacles to trusted computing's widespread use is a high priority for future research. Perhaps the most significant challenge is deploying and managing the PKI necessary to enable general use of the security services that trusted computing supports. Another unresolved issue is standardization. Through the provision of a set of open standards, trusted computing specifies security interfaces that allow heterogeneous devices to interact. Unfortunately, many of the additional technological building blocks required to instantiate a trusted platform aren't standardized, and the TCG doesn't dictate implementation specifics to its adopters. Consequently, several currently available TPMs don't comply with the TPM specifications.[9] The current absence of conformance-testing facilities implies that the production of noncompliant TPMs will likely continue for the foreseeable future. In turn, discrepancies in implementation between TPM manufacturers might limit future interoperability between different trusted platforms.

Many challenges to the successful large-scale use of trusted computing remain, but they're likely to be much less serious for corporate IT users. Providing the full benefits of trusted computing to the widest possible audience is a major challenge for future research. ☐

References

1. P. Gutmann, "PKI: It's Not Dead, Just Resting," *Computer*, vol. 35, no. 8, 2002, pp. 41–49.
2. G. Price, *PKI: An Insider's View (Extended Abstract)*, tech. report RHUL-MA-2005-8, Dept. of Mathematics, Royal Holloway, Univ. of London, 2005.
3. E. Sparks, *A Security Assessment of Trusted Platform Modules*, tech. report TR-2007-597, Dept. of Computer Science, Dartmouth College, 2007.
4. E. Brickell, J. Camenisch, and L. Chen, "Direct Anonymous Attestation," *Proc. 11th ACM Conf. Computer and Comm. Security* (CCS 04), ACM Press, 2004, pp. 132–145.
5. A.-R. Sadeghi and C. Stüble, "Property-Based Attestation for Computing Platforms: Caring about Properties, not Mechanisms," C.F. Hempelmann, ed., *Proc. 2004 Workshop New Security Paradigms* (NSPW 04), ACM Press, 2004, pp. 67–77.

6. J. McCune et al., "Turtles All The Way Down: Research Challenges in User-Based Attestation," *Proc. 2nd Usenix Workshop Hot Topics in Security* (Hot-Sec 2007), Usenix Assoc., 2007; http://usenix.org/events/hotsec07/tech/full_papers/mccune/mccune.pdf.
7. R.O. Sinnott, "Development of Usable Grid Services for the Biomedical Community," *Usability in E-Science Workshop: Proc. Int'l Workshop Interrogating Usability Issues in New Scientific Practice, Within the Lab and Within Society* (NeSC 06), 2006, pp. 26–27.
8. A. Whitten and J.D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," *Proc. 8th Conf. Usenix Security Symp.* (Usenix 99), Usenix Assoc., 1999, pp. 169–183.
9. A.-R. Sadeghi et al., "TCG Inside? A Note on TPM Specification Compliance," *Proc. 1st ACM Workshop Scalable Trusted Computing* (STC 06), ACM Press, 2006, pp. 47–56.

***Shane Balfe** is a research assistant working on a joint MoD/DoD-funded project that looks at security across a system of systems (www.usukita.org) at Royal Holloway, University of London. His research interests include trusted computing, trust management, risk analysis, and ad hoc networks. Balfe has an MSc in information security from Royal Holloway, University of London. Contact him at s.balfe@rhul.ac.uk.*

***Eimear Gallery** is a postdoctoral researcher in the Open Trusted Computing (OpenTC) collaborative project (an EU 6th Framework Integrated Project) at Royal Holloway, University of London. Her main interests include trusted computing and mobile security. Gallery has a PhD in information security from Royal Holloway, University of London. Contact her at e.m.gallery @rhul.ac.uk.*

***Chris J. Mitchell** is a professor of computer science at Royal Holloway, University of London. His main research interests are information security and combinatorial mathematics and its applications. Mitchell has a PhD in mathematics from Westfield College, University of London. He's a member of Microsoft's Trustworthy Computing Academic Advisory Board and the DoCoMo Euro-Labs Advisory Board. He's also a fellow of the British Computer Society (BCS) and the Institute for Mathematics and Its Applications (IMA) and a member of the ACM, the American Mathematical Society (AMS), the IEEE, the London Mathematical Society (LMS) and SIAM. Contact him at c.mitchell@rhul.ac.uk.*

***Kenneth G. Paterson** is a professor of information security at Royal Holloway, University of London. His research interests include cryptography, network security, and trusted computing. Paterson has a PhD in mathematics from Royal Holloway, University of London. He's a member of the IEEE and the IACR, and a fellow of the Institute of Mathematics and its Applications. Contact him at kenny.paterson@rhul.ac.uk.*