Binding Bit Patterns to Real World Entities.

Bruce Christianson
James A. Malcolm

Faculty of Engineering and Information Sciences
University of Hertfordshire : Hatfield
England, Europe

Abstract. Public key cryptography is often used to verify the integrity of a piece of data, or more generally to ensure that operations which modify the data have been requested and carried out by principals who are authorized to do so. This requires keys to be bound to principals in an unforgeably verifiable manner.
Cryptographic bit patterns such as electronic key certificates (EKCs) have a part to play in establishing such bindings, but the requirement ultimately to bind keys to real world entities imposes subtle constraints upon the structure and semantics of EKCs and related entities such as ACLs and capabilities, and upon the role which such entities may play in access control and integrity verification. These do not appear to be adequately realized at present.

1 Introduction

In this position paper, our primary motivation is the problem of ensuring integrity in a wide open system. Enforcement of integrity has two aspects: firstly ensuring that any action which updates the system state is properly authorized (ie ensuring that the right thing is done) and secondly ensuring that the corresponding transaction is properly carried out (ie ensuring that the thing is done right.)

By a wide-open system we mean that network and operating system resources may be shared with strangers, or with other principals whom we do not trust. We also mean that integrity threats may arise from system "insiders" such as our colleagues and system managers, who should therefore be trusted only explicitly, and as little as possible. We do not assume that security domains (ie resources subject to a particular underlying integrity semantics) map neatly onto administration domains: indeed we assume that on some systems resources administered by the same authority may need to be be shared between security domains. (An example is a joint software development project which is contractually subject to certain QA procedures but where one of the subcontractors is also in competition with a different part of the prime contractor on another project.)

In order to check authorization for integrity purposes we frequently need to establish unambiguous verifiable bindings between a bit pattern (such as a name, a cryptographic key, or a program text) and a real-world entity (such as a person, a smart card, or a process running on a particular machine.) The

difference between these two types of entity is that entities of the first type are bit patterns, which can be freely copied and modified in cyberspace; whereas entities of the second type have provenance in the physical world.

Our point of departure is the observation that real world artifacts cannot be authenticated remotely, except by binding them to bit patterns first. Although cryptographic techniques such as digital signature can bind bit patterns to other bit patterns, bit patterns alone cannot suffice to bind bit patterns to unique real world artifacts. A bridge between the electronic world of bit patterns and the "real" world of physical existence must itself have physical provenance as well as containing a representation of the bit pattern. Any attempt to verify the binding between a bit pattern and a real world entity using only electronic certificates is therefore doomed.

2 Binding keys to principals

A public key is a pattern of bits. A principal may be a human, or a piece of software running on a particular system, but in any event a principal is a real-world entity, with properties such as physical location. It is to this real world entity that a key must (eventually) be bound. A binding between a bit pattern and a real world entity cannot be effected solely by another bit pattern such as an Electronic Key Certificate (EKC).

The most an EKC can do is to bind a key to another bit pattern, such as a name or a program text. This is not sufficient to establish the identity (authenticity) of the corresponding real-world entity, since many such entities may "correspond" to the same bit pattern. Bit patterns can be copied, and neither is then the "original". But the binding must indicate which correspondence is intended.

It follows that some real-world artifact is required in order to complete the binding. This could be a copy of the EKC which resides in a particular physically protected location or address space, together with a social context in which the semantics of the protected copy are unambiguous. For example, a legally notarized paper copy of the EKC with established provenance stored in a bank vault would usually do. But the integrity and authentication of electronic access to remote real world entities is problematic, because cryptography protects only bit patterns.

Consequently a primary objective of an access and integrity control system is to reduce to a manageable level the number of times that physical access to a remote real-world entity is actually required in order to resolve a dispute.

One way of doing this is to provide electronic mechanisms which convince honest parties that they are in the right and which convince dishonest parties that recourse to the physical artifacts would in fact resolve the dispute against them.

3 Binding keys to access rights

Access rights must also (at some point) be bound to principals. For our purpose here, we can think of an access right as the capability to invoke a particular transaction (eg to login through a firewall). An access control mechanism based upon signature verification must answer the question: will signature with a key K suffice to satisfy an auditor that a request to invoke transaction T was valid?

It is tempting to bind keys and access rights to principals thus:

(bad) key $\longrightarrow$ principal $\longleftarrow$ capability

The reason this is a bad idea is that it connects two bit patterns via a real world artifact, and thus makes both halves of the link from key to capability impossible to verify electronically. In practice, the principal is usually replaced for verification purposes by a bit pattern which represents its name:

(worse) key $\longrightarrow$ name $\longleftarrow$ capability

In a wide open system, we cannot ensure that such a bit pattern is unique, and therefore the name alone does not suffice to ensure that both key and access right were intended to bind to the same (real world) principal. A better way of doing things is:

(better) principal $\longrightarrow$ key $\longrightarrow$ capability

where of course either binding may be direct or indirect. This is better because public keys are unique by hypothesis: the inability of two parties to generate the same private key independently (whether by accident or design) is part of the foundation of public key cryptography. Of course, there is a chance that A could guess B's key by tossing a lucky coin repeatedly, but this probability can (and should) be made smaller than the chance of A and B being simultaneously struck by a meteor.

4 Indirect bindings

Indirect bindings can be used to reduce the amount of actual change required by updates to access control permissions. One common example of indirect binding is where keys are bound to rights via roles: one table indicates which keys may act in a particular role, and a second table indicates which roles may invoke a particular transaction:

principal $\longrightarrow$ key $\longrightarrow$ role $\longrightarrow$ transaction

The bit pattern representing a role has a similar uniqueness requirement to that for a key. One way of achieving this is to bind a public key of the role manager into the role name.

A similar approach can be adopted to the other half of the verification problem: how should a particular transaction be carried out correctly, and was it? A transaction is typically made up of a number of micro-operations (or method invocations) such as read, overwrite and append, which must be carried out on specific resources for the transaction to be valid:

$$\text{principal} \longrightarrow \text{key} \longrightarrow \text{role} \longrightarrow \text{transaction} \longleftarrow \text{method} \longleftarrow \text{key} \longleftarrow \text{system}$$

Further tables must specify these bindings. To allow for hardware maintenance there may be a further level of indirection mapping virtual resources to physical systems. This process can clearly be continued: the point to note is that the tables (or relation components) representing the inner links of the chains can be treated purely as bit patterns for the purpose of access control, and therefore can be cryptographically protected. Mappings from bit patterns to real-world entities should occur only at the boundaries of cyberspace, not in the middle of the chain.

5 Binding keys to hardware

Internally to the electronic world, real-world principals and systems are represented by keys. How should the bindings at the boundary be effected? A signature is generated in practice by applying a copy of a private key within a particular piece of hardware. Since moving private keys from one piece of hardware to another is a serious security risk, it follows that each key should be permanently bound to a single, identifiable piece of hardware, and physical (rather than cryptographic) means should be used to bind the hardware to specific places or people. (In general, we will need to bind several keys to the same piece of hardware in order to prevent protocol attacks.)

Since principals will use different pieces of hardware from time to time, it follows that a single principal will in general have many keys, which they use to exercise different rights. These rights must be explicitly delegated from key to key, using unforgeable instruments which can be recorded in an audit trail: from the key representing the owner of the resource to the key representing the principal authorized invoke some operation and thence to the key which will certify that the requested operation has been correctly carried out.

It must be possible for such delegation relationships to be verified remotely, and it therefore follows that delegation relationships must be expressed as relationships between bit patterns (keys).

Recent improvements in distributed systems architecture and infrastructure were originally motivated by the desire to allow more freedom over which piece of hardware performed the operations. From the point of view of integrity, however, transparency is a very bad thing, particularly when it extends to the point where examination of an audit trail cannot determine, even in principle, upon which piece of hardware a particular operation took place.

Fortunately, the same mechanisms developed to facilitate transparency can also be used in reverse: the requirement that certain operations be performed only upon particular pieces of hardware where particular keys are located is ceasing to be restrictive.

Relatively rigid bindings between keys and physical objects or locations can be exploited to allow a greater degree of assurance that the operations have been properly authorized and properly carried out, for example by ensuring that malfeasance would require active cooperation between mutually mistrusting administrations.

6 Binding principals to biometric data

Various mechanisms can be used to bind a bit pattern (actually a family of different "acceptable" bit patterns) to a physical person. These mechanisms may use retinal patterns or other vein structures which are only present in living people, and at most one such. These mechanisms are extremely useful, but do nothing to circumvent the need to bind access rights to principals only via cryptographic keys held in trusted physically secure or tamper-evident hardware.

Establishing a binding between a person and their biometric data in the first instance requires physical provenance of the person being certified, except in those cases where we care only that the same person be used on separate occasions, and do not care who they actually are.

Subsequent verification of the match between person and bit pattern requires that trusted hardware be in the vicinity of the person. Remote verification of identity in a wide open system additionally requires that the trusted hardware contain a cryptographic key. So we are again in the familiar word of principals being represented by keys which are associated with hardware which is in some sense trusted.

Adding the biometric data to the audit trail proves nothing: a replay is always possible, since the biometric data itself cannot be kept secret except by hardware which is trusted and tamperproof to begin with. Indeed, if biodata are used instead of keys then separating the different roles used by the same physical person requires an even greater degree of trust in the hardware.

The biometric hardware could be removed from the trust envelope if it were possible to use a living physical person in the same way as an uncopyable secret key. This would require using the person to produce a bit pattern which responds to a challenge and which can be verified without the person present, but which cannot be produced without them. This is a formidable research problem.

In the absence of such an advance, the use of biometric data can at most provide a cheap logical equivalent of a locked room with an armed guard on the door. If A and B are willing to believe the integrity of the hardware used to verify A's biometric data, then A will be willing to allow signature by the key associated with that hardware as a substitute for evidence of his physical presence for the purpose of audit, and B will be willing to delegate rights to the key directly. This then allows all remote verification by third parties (including

audit mechanisms) to take place in the electronic (bit pattern) world. Delegation to remote hardware is expressed as a relationship between bit patterns.

7 Minimizing trust

Clearly a principal should not be empowered to act upon a resource unless the resource owner (or their delegatee) consents. However in a wide open system, the consent of the resource user is equally important: it should not be possible to confer a right unilaterally.

The principle of least privilege says that users should not be given rights they don't need. The obverse principle is that users should not be given rights they don't want. This is the principle of greatest consent : it should not be possible for a principal to acquire a right which that principal has not explicitly agreed to have. The purpose of enforcing this principle is to reduce the need for principals to trust remote systems, by making it harder for a principal to be framed (falsely accused of wrongdoing.) Note that the principle of consent protects the user who doesn't want to trust their system, as well as the server who doesn't want to trust their client.

For example, imagine that B's smart card (reported stolen) is discovered to contain a capability allowing a massive cashpoint withdrawal from an account belonging to the noted public figure A. A claims that B is holding him to ransom. B claims that A has stolen her card and inserted the capability himself. His objective is to blackmail her, using the ransom argument as the threat.

Clearly it is to the advantage of the honest party (whichever they are) that it should be impossible for anybody to place A's capability upon B's card without her prior knowledge and consent.

The principle of consent, combined with the principle of binding keys to hardware, leads to certain consequences: any access right which is bound to a key must be explicitly accepted by the corresponding principal, but the key used to accept a right need not be the key used to exercise the right.

For example, if R is a right, then the right to delegate R is another right R', and the right to accept R is another right R''. The rights R' and R'' may also be delegated from keys to other keys.

Similar remarks apply to the delegation of responsibility: the consent of both parties is required, but the keys which certify this consent may have a very obscure relation ship with the principals.

The verification of the web of delegation should not rely upon knowing who the principals involved are in real life. All interior bindings should be in cyberspace, and in any application where revocation is an issue, there will usually need to be lots of different keys (corresponding to physical places) where the delegation chain can be broken. Once again we see the importance of ensuring that delegation relationships can be verified in the world of bit patterns, without reliance upon knowing the bindings to real-world principals.

8 Auditing

The web of delegation must be recorded and verified by various parties, and this requires an explicit representation of the delegation policy, with guaranteed integrity. Operations must also be properly authorized, and this requires verifiably correct management and distribution of the tables which govern the interior bindings between keys and other bit patterns, and which hence confer access rights. It is important to assure integrity and access control properties for tables such as ACLs, transaction definitions, and role definitions.

Since we have reduced the tables to bit patterns, we can use cryptographic mechanisms to protect them as they move about the system. Access rights to tables are themselves capabilities. Proxies, capabilities, delegation certificates etc can be thought of as small pieces of a (virtual) table in transit. Updates to tables can be treated in the same way as other transactions. For example, unbinding a key from a role effectively revokes that key.

The access control system must be able to convince an auditor that the correct versions of the tables were used to control a given access and that only properly authorized updates to the tables have been made between versions. The audit trail should also enable the parties involved in a series of transactions to determine and undo the damage if the wrong versions were used, and to decide who should bear the cost of this.

Re-entrant application of the same basic access and version control mechanisms used for other shared resources such as files, combined with Optimistic Concurrency Control techniques for recovery, provide a good start for such an integrity mechanism, but the relations between keys and other bitstrings representing roles and rights are subtle.

These relationships need to be encapsulated, passed around a wide open system, verified and audited. It must be possible to make access control decisions which can be justified to an audit mechanism without requiring a prior physical check of the bindings between keys and real world entities in remote parts of the system.

The format and semantics of the next generation of electronic certificates need to be such that end-to-end guarantees can be made which provide all parties with an acceptable basis for proceeding, without covertly requiring that EKCs without secure real-world provenance be treated as having the same status as physical evidence.

The representation and manipulation of delegation relationships will play an important part in achieving this objective. The database community do not attempt to represent real world relationships other than via keys, some of which correspond to real world entities and some of which refer to further entries (relationships) in the database. An analogous approach using cryptographic keys may prove fruitful in the quest for integrity in wide open systems.

9 Conclusions

Real world artifacts have provenance. Real world artifacts cannot be authenticated remotely, except by binding them to bit patterns first. A primary objective of an access and integrity control system is to reduce to a manageable level the number of times that physical access to a remote real-world entity is actually required in order to resolve a dispute.

Bit patterns don't have provenance. Cryptographic techniques such as digital signature can bind bit patterns to other bit patterns, and such bindings can be verified remotely. A binding between a real world entity and a bit pattern cannot be effected solely by other bit patterns, but requires some real world artifact such as a piece of paper or hardware. Any attempt to verify the binding between a bit pattern and a real world entity using only electronic certificates is therefore doomed.

Rights should be bound to keys not to principals. Mappings from bit patterns to real-world entities should occur only at the boundaries of cyberspace, not in the middle of it. Bit patterns should be bound to each other via unique bit patterns and not via real world entities. The only unique bit patterns are those derived from private keys. Trusted hardware is still required.

Keys should be bound to hardware (not to names) and physical means should be used to manage the hardware and to effect the bindings on the boundary of cyberspace. Moving keys from one piece of hardware to another is a serious security risk. Don't do it. Biometrics doesn't reduce the trust requirements for the hardware needed to solve the binding problem, although it does provide a cheap locked room.

Transparency is bad for you, but the good news is that the requirement to do certain things only in certain places is ceasing to be restrictive. Relatively rigid bindings between keys and physical objects or locations can be exploited to allow a greater degree of assurance that the operations have been properly authorized and properly carried out.

Explicit delegation is required both to represent the security policy and to provide an audit trail. Rights may be delegated from keys to other keys. Delegated rights must be explicitly accepted. The key used to exercise the right may not be the key used to accept it. Delegated rights may include the right to accept or to confer other rights, as well as the right to exercise a right or to delegate further the right to exercise it.

Delegation should be expressed as a relationship between consenting bit patterns. The verification of the web of delegation should not rely upon knowing who the principals involved are in real life. If revocation is to be supported, it must be easy to break the delegation chain in lots of places.

An EKC is best thought of as a little piece of an ACL in transit. Electronic certificates cannot be adequate substitutes for the paper versions, and should not try to be. Both are required: their purpose is different.

References

1. Ellison, C.M., 1996, Establishing Identity without Certification Authorities, Sixth USENIX Security Symposium Procedings 67–76
2. Ellison, C.M., Frantz, B. and Thomas, B.M., 1996, Simple Public Key Certificate, http://www.clark.net/pub/cme/
3. Harbison, W.S., 1997, Trusting in Computer Systems, PhD thesis, Computer Laboratory, University of Cambridge
4. Lek, H. van der, Bakema, G.P. and Zwart, J.P.C., 1992, De Unificatie van Objecttypen en Feittypen een Practisch en Diadactisch Vruchtbare Theorie (Unifying Object Types and Fact Types : A Practically and Didactically Productive Theory), Informatie **34**(5) 279–295
5. Low, M.R. and Christianson, B., 1994, Self Authenticating Proxies, The Computer Journal **37**(5) 422–428
6. Needham, R., 1997 The Changing Environment for Security Protocols, IEEE Network **11**(3) 12–15
7. Rivest, R.L. and Lampson, B., 1996, SDSI - A Simple Distributed Security Infrastructure, http://theory.lcs.mit.edu/~rivest/
8. Snook, J.F., 1992, Towards Secure Optimistic Distributed Open Systems, PhD thesis, University of Hertfordshire : Hatfield
9. Roe, M., 1997, Cryptography and Evidence, PhD thesis, Computer Laboratory, University of Cambridge

This article was processed using the LaTeX macro package with LLNCS style