2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing

A Domain-Oriented Approach for Access Control in Pervasive Environments

Jun Li
Security & Privacy
Enterprise Risk Service
Deloitte, London
United Kingdom EC4A 4TR
joexli@deloitte.co.uk

Bruce Christianson
School of Computer Science
University of Hertfordshire
Hatfield, Herts
United Kingdom AL10 9AB
b.christianson@herts.ac.uk

Abstract

Pervasive computing envisions an environment in which we are surrounded by many embedded computer devices. Those networked devices provide us with a mobile, spontaneous and dynamic way to access various resources provided by domains with different security policies. The conventional approach to secure access over multiple domains is to implement a universal trusted infrastructure, extending local identity- or capability-based security systems and combining them with cross-domain authentication mechanisms. However, this does not adequately meet the security requirements of communicating with strangers in pervasive environments. This paper presents an intrinsically multi-domain oriented approach which incorporates an identity-based encryption (IBE) access control mechanism. This approach allows the right domain to get involved with its local players' interactions by helping them to convert a token to a usable access capability, whilst facilitating revocation.

Keywords: Domain-oriented access control, pervasive computing, identity-based encryption, revocation.

1. Introduction

Today's spontaneous and highly decentralised pervasive computing environment [24, 25, 29] has raised a number of new challenges in a considerably under-explored territory, *security over multiple domains*. By the term *domain*, we denote the scope of security policy rather than geographic location. Traditionally, security in the multiple domain context has not been holistically researched. A major argument is that a successful protocol targeting one domain can "easily" be lifted to multiple domains. This is mainly fulfilled by requiring assistance from a globally trusted infrastructure.

However, in pervasive environments, resources (either hardware, e.g. a DVD player, a fax machine, a wireless camera, or software programs running on various devices) are usually provided by a variety of different suppliers, for instance, organisations, companies, universities, shops, even other human users, and so on. Thus, there is typically a lack of a global infrastructure which every player will trust. Moreover, we do not desire to force each domain to implement the same (or even compatible) security infrastructure and mechanisms.

Consequently, we have proposed an architectural framework, Localisation of Trust (LoT) [21], to deploy domains as a primary setting of pervasive environments [1]. We argue that security for the pervasive environment can be solved better, if security for multiple domains is solved first. This paper provides a conceptual discussion of our approach for access control in multiple domains. We also give an existence proof, in the form of a Kerberos-like protocol based on Identity-based Encryption (IBE), which supports the approach we advocate. Further protocols and mechanisms are discussed in the research report [21].

This paper starts with a brief review of previous approaches. In section 3, we highlight the security need of talking to the right strangers in pervasive environments, and introduce the notion of *localising the trust*, followed by a description of our domain-oriented approach in section 4. An architecture for Encryption-Based Access Control (EBAC) and the important revocation issue will be explained in section 5 and 6 respec-

[1] The term *pervasive environments* in this paper includes (but is not limited to) any pervasive computing applications that *genuinely* involve many different domains.

tively. We then illustrate a simple pervasive computing example using domain-based approach in section 7. Finally, conclusions are set out in section 8.

2. Previous Approaches

Efforts to secure pervasive computing and fundamental communication structure (mobile ad-hoc networks) have received increasing attention in the security research field. In their novel "Resurrecting Duckling" security model [26, 27], Stajano and Anderson mark a *physical contact* idea to bootstrap trust between strangers. Moreover, based upon this approach, Balfanz *et al.* [4] introduce a pre-authentication process to exchange some relevant cryptographic material in a demonstrative identification (physical recognition), authentic and secure location-limited channel (such as infrared, ultrasound or a short-wire).

Kagal et al. [18, 19] target the security challenges which have arisen from lack of central control and rarely predetermined users in the pervasive computing environment. Their systems are built from XML language to form *distributed trust* (following Blaze's PolicyMaker [5]) rather than just user authentication and access control. To access a resource, a foreign user Bob requests permission from a local user Alice. Alice hands over to Bob a signed delegation certificate that implies the proper access rights are delegated to Bob. Then, Bob's request will be granted after he sends the security manager this delegation certificate along with his identity certificate. The trust-based systems will make the final decision based upon the policy check, e.g. whether Alice's rights are revoked.

Among classic capability-based access control system, Gong's I-CAP [15] is designed keeping multiple domains in mind. Potentially, the domain in an I-CAP system is not geography-based. It introduces two forms of capabilities, external and internal capabilities. The resource server generates a random number R_0 for a resource $Object$. The internal capability ($Object$,R_0) is only known by the server. Then, the servers computes R_1, which is the hashed value of the group (ID_{user},$Object$,$AccessControl$,R_0) for a local user in the system. The external bit-pattern (ID_{user},$Object$,$AccessControl$,R_1) is delegated to the corresponding user. As a result, different users hold different bit-pattern which corresponds to capabilities for the same object. Unfortunately, the I-CAP system requires a good underlying (cross-domain) authentication mechanism to address the freshness problem.

3. Paradigm Shift in Security

If a pervasive computing environment only replaces wires with wireless RF media, then to secure such an environment is not too hard, considering the well developed cryptosystems. However, the new challenges [9, 11, 27], for instance, poorly defined network boundaries, dynamic enrollment, no pre-configuration, transient association and decentralised infrastructure, have entailed a massive qualitative change in security requirements.

3.1. Talking to the Right Strangers

An interesting threat emerging from pervasive computing environments is, talking to *incorrect* strangers, who are usually from different domains and not likely to pre-establish secure knowledge (e.g. crypto-key information) or trust relationships. An old paradigm to address this problem is to create a trusted environment by employing a global unique *trusted infrastructure*, such as Kerberos [28], PKI, etc. They are introduced to help players to determine correct strangers usually by verifying their IDs, names, long-term public keys, or roles. Thus, a security system or protocol implemented successfully in one domain can be migrated to different domains involved in the communications, combining with both important and necessary cross-domain authentication mechanisms.

In pervasive environments, however, we have no prior knowledge about the names/IDs, or roles of those to whom we are going to talk, or which kind of privilege a user needs to access resources. It is not only infeasible to create a trusted environment to which those players can connect: more importantly, the need to have global trust is actually distracting us from what is more important in our own domains. The trusted infrastructure has its required assumptions. Thus, inevitably, players have to set up their own assumptions and security policies in accordance with those of the infrastructure. However, the semantics of the threat model will be different due to the nature of pervasive environments. Threats are specific to the requirements from different pervasive applications and assumptions which will not be known to the infrastructure. Security thus becomes a harder problem as soon as a user is *required* to trust an *arbitrary* external authority. Consequently, we should not let infrastructure dictate local policy, and we should avoid building anything on top of cross-domain authentication.

3.2. Localising the Trust Security Principle

When an interaction crosses domain boundaries, it is not only hard but (we shall argue) also unnecessary for a player from one domain to understand the precise security policy and mechanisms from other domains. Thus, we need a different [2] security design principle to guide security policy establishing trust for pervasive environments. The main security design guideline in security policy in LoT is the *localising the trust* principle. This allows a pervasive player to put trusted things in her own domain, or some places that the player already has stable connection with. Explicitly, domains are responsible to make the security decisions for their local players with respect to local policies.

In fact, the localising the trust principle ties a player's claim closely with her affiliated local domain, encouraging domains to get involved with their local players' interactions. Hence, we need to have a form of information showing which domain the player is from[3], like an email address. An email address clearly indicates the association between a domain and a player, for instance, *LJ@herts.ac.uk*, which contains the knowledge that "the user *LJ* is coming from domain *herts.ac.uk*". It is another issue to verify the authenticity of the domain, "is there really a domain named *herts.ac.uk* (and is it the university or not)" or the relationship, "is *LJ* really AT *herts.ac.uk*". Whether such an association between users and domains actually exists is controlled by the domain-oriented approach described in the next section.

4. Domain-Oriented Approach

In our work, the concept of achieving access control in the multiple domain context is *syntactically* similar to the one witnessed in conventional environments. *Semantically*, however, the domain-oriented approach for access control in pervasive environments differs by allowing domains to authenticate their local players. Hence, the difficult problem (access control over multiple domains) is reduced to two considerably easier problems (user authentication in a single local domain and remote authorisation between two domains) that we are more confident to deal with.

This is more efficient than implementing authentication across domains. As illustrated in figure 1, it is Bob's job to convince domain B that he is entitled to access this particular resource. Alice has delegated some rights over her resources to players from domain B. Alice does not care how Bob authenticates himself to his own domain. She is happy to grant Bob's access once the conversion is completed successfully.

[2] *Different* in the sense of changing the way of thinking on current security policies. The policies themselves may be perfectly adequate in some cases.

[3] For readers who are interested in this, the LoT framework [21] actually introduced a novel Profile Certificates to make this association.

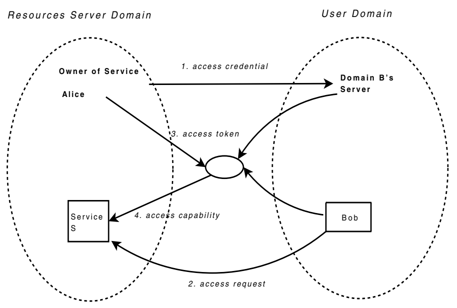

Figure 1: Alice delegates a domain-based access credential (AC_B) to a user domain B (i.e. its domain server). When a player Bob from domain B requests to access a particular resource owned by Alice, she will give Bob an access "token" based upon Bob's access request. This "token" can only be converted to a usable access capability help from the correct domain B.

4.1. Remote Authorisation - Delegation of Rights

Delegation is a natural consideration for making access control decisions [1]. Thus, in the domain-oriented approach, the resource domain intends to *delegate* some access rights (on a particular resource) to other users' domains, very often with some restrictions. Usually, Alice generates some form of *access credentials*, depending upon the delegation mechanisms deployed within her own domain, and hands over them to domain B.

Those access credentials are required to be:

- **Domain dependent**: credentials should be domain relative. It is important that different user domains should have different bit patterns of *access credentials* for the same access right on the same resource. In this way, the compromise of one user domain does not help an attacker to gain any information of usable forms of access credentials for other user domains. Moreover, the delegation mechanism deployed in the resource domain needs to ensure that the collusion of some misbehaving user domains still cannot assist the attacker in terms of constructing a usable access credential.

- **Presentation restriction**: in contrast with most delegated access control systems (where possession of a delegated access credential is both necessary and sufficient to gain access, e.g. a secure capability), the (direct) presentation of the domain-relative credentials is neither necessary nor sufficient for a player to access a certain resource. Actually, post-issue presentation is extremely restricted. Alice will not grant access to Bob if he naively submits domain B's access credential. Instead, Alice will regard domain B as compromised and revoke domain B's access right, because domain B must explicitly delegate the credential to Bob.

An example of this type of access credentials is dual capabilities described in the LoT framework [21]. It heavily borrows Gong's novel idea of having internal and external capabilities. However, the basic system infrastructure is built from the domain-oriented approach for access control. Instead of requiring an additional cross-domain authentication mechanism to check the user's ID at the time of using the external capability, authentication occurs only locally within the user's own domain. As a consequence, our access credentials are not simply access control *tokens*. In fact, their primary function is to let (remote) users authenticate themselves to their own, and more importantly *correct*, domain.

4.2. Localised Authentication

In our domain-oriented approach, the authentication step only occurs locally (within the user's own domain). Hence, from Alice's perspective, it is Bob's problem to authenticate himself/herself to the correct domain B. This can be realised by many existing mechanisms, e.g. classic Kerberos-like protocols [12, 28], policy-based trust management [5], or more pervasive security protocols [4, 11, 27], etc. We do not discuss those mechanisms in this paper. Instead, we encourage domains to choose a proper authentication mechanism freely and be responsible for their choice, in accordance with their own domain policies.

The basic idea of *localised authentication* has been hinted in the Identity-based Encryption cryptosystem (IBE). The IBE system does not require a chained or transitive trust relationship along the transmission path. Instead, trust is only established between end-users and a local trusted party, e.g. Private Key Generator (PKG) in the user's local domain. But to our knowledge, the use of IBE for *access control* has not previously been significantly explored. We desire a mechanism connecting remote authorisation and localised authentication steps in the domain-oriented approach for access control over multiple domains. This can be realised by an encryption-based access control mechanism.

5. The Architecture of Encryption-based Access Control (EBAC)

From the perspective of encryption, the encryption-based access control sketched here is analogous to some existing Identity-based encryption (IBE) schemes [2, 6, 13, 17]. For instance, it lets Alice encrypt a message in a way that *Bob* can only decrypt with the necessary assistance from some security services in his own domain (domain B). In EBAC, this message will be an access capability for a resource in Alice's domain.

From our perspective, pairing-based IBE is not suitable for the multiple domain context. In those systems, every domain's public key has to be available, at least at the time when a user requests access. If the authenticity of the domain's public key is *certified* by some external certificates authorities, the certificate revocation and trust transitivity problems re-emerge.

The basic construction for encryption-based access control proposed in this paper is converted from Goldreich *et al.*'s self-delegation scheme [14]. In their original system, the purpose is to delegate certain rights from a user to a user himself without risking the compromise of her long-term private/public key pair (primary public/private key). Accordingly, secondary key pairs (sk_ℓ, pk_ℓ) are created by the user. They can only be validated with a validation tag (val_ℓ) based upon a certain limitation (the limitation index ℓ). Given a triple (sk_ℓ, val_ℓ, pk_ℓ), the player is able to convince a verifier that a certain public key pk_ℓ can be used on behalf of the primary key (given the limitation index ℓ, a primary public key and the necessary system set up parameters).

We take advantage of the fact that a private/public key pair can also be applied to do decryption/encryption operations. Assume that PK^* is the full encryption key and SK^* is the full decryption key.

An overview for the EBAC scheme described above is illustrated in figure 2. Alice generates a master secret SK firstly in her own domain and computes a pair (SK_ℓ,PK_ℓ) for a user domain B. This pair is called the *access rights pair*, where PK_ℓ is AC_B. From the requirements of being *domain-dependent*, we know,

- Given a set of values from a set of PK_ℓ, it is still (computationally) infeasible to compute any SK_ℓ or the master secret value SK.

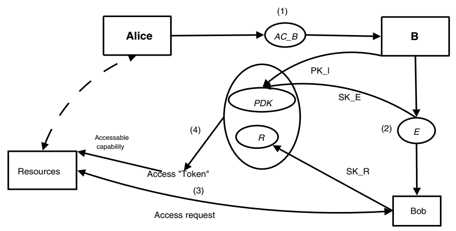

Figure 2: to access Alice's resources, a stranger Bob has to retrieve a partial decryption key (PDK) from his own domain B.

The construction of the access rights pair will be highly dependent on the mechanism that Alice chooses to use to delegate access rights, e.g. a secure capability, a public key pair, and so forth.

Encryption/Decryption: now, we will have,

- The Partial Decryption Key (PDK) which can be provided by domain B and is based upon the access credential (AC_B) delegated from Alice to domain B during the *remote authorisation* step. Given any collections of issued $PDKs$ from a set of local players in domain B, an attacker cannot compute the value of AC_B.

  $$PDK = AC_B \cup SK_\epsilon = PK_\ell \cup SK_\epsilon, \textit{ (1)}$$

  SK_ϵ is added to AC_B to generate a PDK. SK_ϵ is the secret part of the key pair (SK_ϵ, PK_ϵ) generated by domain B after inputting the access index ϵ. The use of SK_ϵ is to guarantee that the AC_B will be kept secret to domain B (the user's domain). The key pair (SK_ϵ, PK_ϵ) is called the *Endorsement Pair* in EBAC.

- A full encryption/decryption key pair,

  $$\text{Encryption key: } PK^* = SK_\ell \cup PK_r \cup PK_\epsilon, \textit{ (2)}$$
  $$\text{Decryption key: } SK^* = PDK \cup SK_r = PK_\ell \cup SK_\epsilon \cup SK_r, \textit{ (3)}$$

  Based upon the idea of *delegation of responsibility* [10], EBAC lets Bob generate a key pair (SK_r, PK_r) and commit the secret component SK_r. This plays two important roles here. First of all, the existence of SK_r prevents a misbehaving domain B to masquerade as *Bob*. Secondly and more importantly, if *Bob* decides to collude with another local player Moriarty from the very beginning, *Bob* has to give SK_r to Moriarty. This will force Bob to compromise his personal secret to an attacker, in order to breach security. We can consider for example that this (SK_r, PK_r) pair is associated with some form of digital cash [3, 7, 23] for Bob. Thus, Bob is not willing to give up SK_r by any means. The key pair (SK_r, PK_r) is called a *Responsibility Pair* in EBAC.

- Two algorithms, *Encrypt*{} and *Decrypt*{}.

  1. *Encrypt*{}: this algorithm is used to encrypt a message M under the encryption key, PK^*, after inputting SK_ℓ, PK_r, PK_ϵ.
  2. *Decrypt*{}: correspondingly, it is called by Bob to recover the message M by taking the full decryption key, SK^*, after inputting PDK and SK_r.

Note that "$\cup$' here is only a symbol. The mathematical meaning varies depending on the underlying cryptographic algorithm chosen in the implementation (again, more practical approaches based upon discrete-logarithm can be found in our work [21]).

Generally speaking, revocation works closely with access control, the delegation semantics in particular. We cannot discuss *delegation* without having *revocation* in mind.

6. Revocation

Revocation is one of the main difficulties for many security systems. A notorious case is the (public key) certificate revocation problem that has been witnessed in many PKI-based systems. Conventionally, the revocation problem has been dealt with separately and some ad-hoc mechanisms have been provided for managing revocation information [16]. However, it is difficult to believe that players will understand and check these detached revocation mechanisms at all. Also, it is problematic to support the *timeliness* of revocation information. Particularly, since pervasive communications usually involve multiple domains, a domain may not always be aware of security policy changes in other domains. In addition, the extra cost to manage revocation will be significant because most pervasive applications may take place on a purely temporary basis.

The domain-oriented EBAC brings a number of significant impacts (i.e. immediateness, selectiveness and revocation transitivity) by integrating revocation as part of a normal transaction. If domain B wants to revoke a local user Bob's access rights on accessing shared resources owned by Alice (which may be long-term, e.g., Bob is not an employee any more, or may be temporary invalidation, e.g. Bob takes one day's leave), domain B needs only to stop issuing the PDK, when Bob requests the necessary crypto key materials in real-time interactions. As shown in figure 2, Bob cannot generate a full decryption key without acquiring a *current* PDK from domain B. Moreover, Bob is not able to compute the correct PDK on his own. Consequently, the PDK can be considered as the *revocation factor* created by domain B. Once B drops its pointer to the PDK, the access request from Bob immediately stops forwarding. All the access statements Bob has become useless.

Alternatively, Alice may not want any players from domain B to access a certain resource any more, for instance, the contract expires. She just needs to revoke the proper access rights for domain B. Since revoking access rights takes place within her own domain, Alice can maintain a domain-based access control list *locally* for instance [4]. If she wants to revoke *a domain's* access rights, she deletes this domain's access credentials from the *domain access control list*. To compose a local player's access credential, the domain's one has to be included. As a result, if a request contains a revoked domain's information, Alice will abort the communication.

7. A Pervasive Computing Scenario

Consider a simple but interesting (in the security protocol sense) pervasive application, the public meeting. Alice is the marketing manager of company A, and she meets company B's marketing manager Bob for the first time in a public conference room. Alice would like to securely transfer a private project plan m from her personal device (DRD_{alice}) to Bob's handheld device (DRD_{bob}). Assume Alice does not know *Bob* in person, so in other words, *Bob* is a stranger to Alice although company B is known to Alice. Thus, Alice has to be convinced that she is talking to the right *Bob* (in addition to talking to the right device).

In most conventional approaches, for instance, Kerberos, Bob has to get a ticket from company B's server before this meeting. Then, he submits this ticket to Alice during the meeting. If Alice recognises Bob's ticket, she checks the access rights associated with the ticket to determine if Bob has the correct rights (compared with his access request) to access the document m. If so, Alice grants Bob's access request. Otherwise, Bob's access will be refused.

For our domain-oriented approach, briefly speaking, Bob needs to submit his access credential, but this is not used for Alice to make a final access decision. Instead, it contains just sufficient information to tell Alice which domain Bob is from. Alice checks whether company B already has permission to access the document m. If so, Alice gives Bob an encrypted access token which can be converted to a useful access capability only by the correct domain, i.e. company B. This conversion can be completed by the credentials (e.g. dual capabilities) established during the remote authorisation stage between two companies. Most importantly, Bob can only decrypt this token by authenticating himself to his own company in accordance with his own company's policy. If company B is willing to delegate Bob access to m, company B's server will help Bob to decrypt the token. If company B does not want Bob to access the plan m, Bob cannot decrypt the token. Thus, Bob's access is granted only if the token issued by Alice is converted by Bob's domain to a useful access capability. As well as being authenticated by his own domain, Bob must also prove to Alice that he is in control of the hardware to which Alice has given the token [5].

This scenario demonstrates the effectiveness and efficiency of deploying a domain-based approach to address the access control issue in pervasive environments. Essentially, Alice does not care who Bob is as long as Bob's domain, i.e. company B, has the correct access credentials and Bob has his own domain's permission to use them.

8. Conclusions

A powerful tool to address access control in pervasive environments is to understand pervasive computing environments from a *multi-domain* perspective. In this approach, the final commit/abort security decisions are ultimately managed by local domains rather than external authorities. This is an efficient approach because a player does not need to understand the semantics of security mechanisms in other domains. Moreover, by treating revocation as part of the normal transaction, the EBAC scheme effectively solves the revocation bottleneck inherent in most delegation-focused access control systems.

References

[1] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A Calculus for Access Control in Distributed Systems. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 15(4):706 – 734, Sept. 1993.

[4] Having a *local* access control list is not a step backward. Like Karger's S-CAP [20], an access control list is adequate here as long as it can be kept *local* to Alice's own domain.

[5] Please refer to our earlier works [22, 8] on two-channel authentication protocols for achieving talking to correct devices in pervasive environments.

[2] S. Al-Riyami and K. Paterson. Certificateless Public Key Cryptography. In C.S.Laih, editor, *Proc. ASIACRYPT 2003*, LNCS 2894, pages 452 – 473. Springer, 2003.

[3] R. Anderson, C. Manifavas, and C. Sutherland. Netcard – A Practical Electronic Cash System. In *4th Cambridge Security Protocols Workshop*, 1996.

[4] D. Balfanz, D. Smetters, P. Stewart, and H. Wong. Talking to Strangers: Authentication in ad-hoc Wireless Networks. In *Symposium on Nework and Distributed Systems Security (NDSS'02)*, Feb. 2002.

[5] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. In *Proc. IEEE Conference on Security and Privacy*, pages 164–173, Oakland, CA, May 1996.

[6] D. Boneh and M. Franklin. Identity based Encryption from the Weil Pairing. In *Advances in Cryptology - ASIACRYPT 2001*, pages 213–229. Springer-Verlag, 2001.

[7] D. Chaum, A. Fiat, and M. Naor. Untraceable Electronic Cash. In *Advances in Cryptology - Crypto '88*, pages 319 – 327. Springer, 1990.

[8] B. Christianson and J. Li. Multi-channel Key Agreement using Encrypted Public Key Exchange. In *15th Security Protocols Workshop*, Apr. 2007. To appear.

[9] S. Creese, M. Goldsmith, B. Roscoe, and I. Zakiuddin. Authentication for Pervasive Computing. In D. Hutter et al., editor, *Security in Pervasive Computing 2003*, LNCS 2802, pages 116 – 129. Springer-Verlag Berlin Heidelberg, 2004.

[10] B. Crispo. *Delegation of Responsibility*. Ph.D thesis, Wolfson College, the University of Cambridge, May 1999.

[11] L. M. Feeney, B. Ahlgren, and A. Westerlund. Spontaneous Networking: An Application-Oriented Approach to Ad Hoc Networking. *IEEE Communications Magazine*, pages 176–181, June 2001.

[12] A. Fox and S. D. Gribble. Security on the Move: Indirect Authentication using Kerberos. In *Procceedings of the 2nd Annual International Conference on Mobile Computing and Networking*, pages 155 – 164. ACM Press, 1996.

[13] C. Gentry. Certificate-based Encryption and the Certificate Revocation Problem. In E. Biham, editor, *Proc. EUROCRYPT 2003*, LNCS 2656, pages 272–293. Springer, 2003.

[14] O. Goldreich, B. Pfitzmann, and R. L. Rivest. Self-delegation with Controlled Propagation — or — what if you lose your laptop. *Lecture Notes in Computer Science*, 1462:153–168, 1998.

[15] L. Gong. A Secure Identity-based Capability System. *IEEE symposium on security and privacy*, pages 56–65, 1989.

[16] C. A. Gunter and T. Jim. Generalized Certificate Revocation. In *POPL '00: Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 316–329, 2000.

[17] U. Hengartner and P. Steenkiste. Exploiting Hierarchical Identity-Based Encryption for Access Control to Pervasive Computing Information. In *Proc. of SecureComm 2005*, pages 384–393, Sept. 2005.

[18] L. Kagal, T. Finin, and A. Joshi. Trust-based Security in Pervasive Computing Environment. *Computer*, 34(12):154 – 157, Dec. 2001.

[19] L. Kagal, J. Undercoffer, F. Perich, A. Joshi, T. Finin, and Y. Yesh. Vigil: Providing Trust for Enhanced Security in Pervasive Systems. Technical report, University of Maryland Baltimore County, Aug. 2002.

[20] P. A. Karger. *Improving Security and Performance for Capability Systems*. Ph.D thesis, University of Cambridge, 1988.

[21] J. Li. *Toward a Localisation of Trust Framework For Pervasive Environments*. Ph.D thesis, University of Hertfordshire, Mar. 2008.

[22] J. Li, B. Christianson, and M. Loomes. Fair Authentication in Pervasive Computing. In *Secure Mobile Ad-hoc Networks and Sensors (MADNES'05)*, volume LNCS 4074, pages 132 – 143. Springer Berlin/Heidelberg, Aug. 2006.

[23] R. L. Rivest and A. Shamir. PayWord and MicroMint: Two Simple Micropayment Schemes. In *4th Cambridge Security Protocols Workshop 1996*, pages 69 – 87, 1996.

[24] D. Saha and A. Mukherjee. Pervasive Computing: A Paradigm For The 21st Century. *IEEE Computer*, 36(3):25 – 31, Mar. 2003.

[25] M. Satyanarayanan. Pervasive Computing: Vision and Challenges. *IEEE Personal Communications*, (10-17), Aug. 2001.

[26] F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In B. Christianson, B. Crispo, and M. Roe, editors, *Security Protocols, 7th International Workshop Proceedings, Lecture Notes in Computer Science*, LNCS 1296, pages 172 – 194, 1999.

[27] F. Stajano and R. Anderson. The Resurrecting Duckling: security issues for ubiquitous computing. *IEEE Computer*, 35(4):22–26, Apr. 2002.

[28] J. Steiner, C. Neuman, and J. Schiller. Kerberos: An Authentication Service for Open Network Systems. In *Proceedings of the Winter 1988 Usenix Conference*, Feb. 1988.

[29] M. Weiser. The Computer for the Twenty-First Century. *Scientific American*, 265(3):94 – 104, Sept. 1991.