FLEXIBLE DELEGATION SECURITY FOR IMPROVED DISTRIBUTION IN UBIQUITOUS ENVIRONMENTS

Georgios Kalogridis[1], Chan Yeob Yeun[1] and Gary Clemo[1]
[1]*Toshiba Telecommunications Research Laboratory, England*
{georgios.kalogridis;chan.yeun;gary.clemo}@toshiba-trel.com

Abstract: Since one of the services that delegation offers is a more optimised distribution paradigm, reconfigurable terminals are envisaged to be able to show how security can negotiate and adapt itself in a flexible way, in favour of performance, without nevertheless compromising fundamental security requirements. So far we observe that protocols that enable cascade delegation and maintain end-to-end accountability among all the involved actors, demonstrate heavy computations or increased network usage and unnecessary redundant complexity. With a novel set of delegation algorithms we show that it is possible to maintain end-to-end accountability and at the same time have compact and lightweight cascade characteristics. This has been further enhanced with an innovative security mechanism that can be automatically optimised in various environments. Due to these attributes, the proposed delegation algorithms will additionally enable reconfigurable terminals to perform complicated tasks either when they enter an untrusted PAN or they use a slow communications link. These characteristics make our protocols suitable for ubiquitous environments as well as for diverse personalised services and mobile applications.

Key words: Flexible security, cascade delegation, multicasting delegation, ubiquitous distribution

1. INTRODUCTION

The main goal of this paper is to introduce systems that will be able to adapt some of their security mechanisms in a flexible way in favour of performance without nevertheless compromising sensitive security requirements. Distributed and wireless security considerations are mainly

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35691-4_52 focused on delegation techniques which are considered to be a mechanism able to leverage richer mobile services for the benefit of the user. Personal Area Networks (PAN) are considered to be promising environments that will enable diverse applications to be realised in mobile devices in the future. These applications will mainly be highly distributed and in many cases services will dynamically be downloaded to terminals.

Delegation techniques so far either restrict themselves to specific systems and networks or do not present robust and efficient characteristics. In this paper we attempt to introduce a set of protocols that will enable delegation in diverse networks and will be suitable for a wide variety of services. At the same time we strive to reduce the complexity and specialised infrastructure dependencies. Our proposed delegation will be able to configure itself in a private or public PAN or in complex heterogeneous network topologies. In order to completely support ubiquity, our protocols will adapt in a dynamic environment by exploiting a proposed trust evaluation mechanism.

2. CONCEPTS

2.1 Delegation Background

The value of delegation in distributed environments is that it can significantly enhance the efficiency of usage of local or remote resources and provide end users with more qualitative services and richer applications. A delegation service could save a mobile terminal from expending large amounts of CPU, network interaction, battery as well as tedious and lingering human interactions with the terminal and the network. However, the introduced security implications impose quite challenging problems.

Various general-purpose delegation protocols that have been proposed so far seem not to be suitable for ubiquitous and diverse services. Some are either cryptographically heavy or do not provide robust accountability. A good introduction to various delegation schemes can be found in [4].

Cascade delegation described in [8] suggests quite efficient and secure protocols. However the requirement of authentication is met with an increased complexity and the usage of an authentication server. Similar techniques are used in [9] where either nested or chain delegation tokens are used, however tracing and accountability are not straightforward. In [3] fine grained levels of accountability and authentication are introduced. This offers optimum security but it has an undesirable effect on the performance, since it needs excessive network and computing resources.

Finally in [11] a very efficient delegation protocol is presented, ideal for terminals and capable to tackle demanding mobile agent applications that might include secure software download [10] or intelligent, personalised and localised mobile commerce. In this proposal cascade delegation is more compact and is suited for reconfigurable terminals and PAN applications.

2.2 Problems and Approaches

One main problem that delegation protocols [1], [3], [5], [8], [9] suffer from is unnecessary dependencies between authentication and delegation. This is not recommended because key management becomes cumbersome and there is no clarity for accountability of actions. On the other side of the spectrum, protocols that use role-based asymmetric or symmetric keys introduce high complexity. These issues have been tackled successfully in [11] by using only one message exchange from the grantor to the grantee.

On the other hand, delegation in [11] cannot support simultaneous migration to various mobile agents in a multicasting way. The success of the one message exchange depends of the usage of a short-lived delegation token and as a result a delegation broadcast in parallel is not applicable. Although this might not be a practical problem in certain networks, it is nevertheless envisaged that in future ubiquitous networks multicasting delegation might be required. In addition the protocol demands the terminal to both create the delegation keys as well as sign and encrypt the delegation token, which offers maximum security but has an impact on efficiency.

In this paper we will address the above issues and further improve current delegation protocols. This will be achieved with novel end-to-end design from the grantor to the final server and through the whole chain of delegates. More specifically we introduce for first time a kind of flexibility and multi-mode operation that will support ubiquitous delegation in different network configurations (PAN, Internet, ad-hoc, cellular) or a combination of them.

The main obstacle that we have to overcome is how to migrate the same delegation in different networks and how to leverage this with a compact mechanism. In order to achieve that, we have to facilitate dynamic selection of mode of operation in a single cascade delegation. In addition the problem of extending a terminal's capabilities in a multi-domain distributed environment is taken under consideration. With our novel approach we try to tailor the cryptographic burden to the specific environmental requirements but we never compromise fundamental security requirements.

2.3 Notation

Entities within the distribution system are denoted with capital letters. The principal that delegates will be denoted with the A_i letter and the principal that will perform the final transaction will be denoted with the B

letter. Between A_1 and B there will always be A_2 that receives delegation from A_1. A_2 might further transfer some delegation rights, if applicable, to A_3 and so on. At the end an A_f will exercise the delegation rights by contacting B. At every single step the grantee will have to form a Delegation Token. The Delegation Token that A_i gives to A_{i+1} will be denoted as DT_i.

In addition, we denote SK_i and VK_i respectively as the Signing Key and the Verification Key that belong to A_i. Next, we denote DSK_i and DVK_i to be the Delegation Signing and the Delegation Verification keys respectively. This is a separate asymmetric key pair that A_i will use for delegation purposes. This will be created and maintained by A_i, but A_{i-1} will have to authorise it in order to give A_i the power to exercise roles that are defined in DT_{i-1}. This key pair does not lie within any network of trust and only A_i is responsible for keeping DSK_i private and sharing DVK_i.

At this point, we have to make clear that when asymmetric keys are used standalone then they indicate the actual keys but when they are followed by brackets *(data)* then they represent the actual cryptographic function that will either sign or verify the enclosed data. We also suggest that signing functions are performed on hash values. Hence $DSK_i(data)$ or $SK_i(data)$ will more explicitly represent $DSK_i(h(data))$ or $SK_i(h(data))$ respectively.

Finally, in equations (1), (2), (3) we define the Signed Delegation Token (SDT_i) that A_i transfers to A_{i+1}, a fresh random sequence and a series.

$$SDT_i = r_i \| SK_i(A_i \| A_{i+1} \| DT_i \| r_i \| DVK_{i+1}) \quad (1)$$
$$r_i = T_i \| N_i \text{ , or } r_i = T_i \text{ , or } r_i = N_i \quad (2)$$
$$C_i(a_{i-k}) = a_1 \| a_2 \| \cdots \| a_{i-k} \quad (3)$$

With || we symbolize concatenation. The fresh random sequence, r_i, represents a unique piece of information that could be either T_i, which is a time-stamp, n_i, which is a nonce, or a combination of both of them. As a nonce either a sequence number or random data could be used. It should be mentioned that r_i is produced by A_i, just before it creates and sends the SDT_i and it will be always bound to that. In any other cases r_i' will be used.

2.4 Design Issues and Architecture

In our delegation protocols the primary concerns are provision of authentication, integrity and accountability. Accountability is actually formal evidence that a specific actor made a specific action. Accountability of delegation enables detection of a fraudulent delegate that does not strictly comply with the delegated policy. Accountability will also enable an innocent actor to protect itself from allegations of misusing its rights.

For robust accountability, it is strongly recommended that the Delegation Token (*DT_i*) with include the following entries:

- An exact formation of sets of tasks to be delegated from a specific entity to another specific entity. Such a format has been proposed in [2].
- Indispensable component of every *DT_i* should be an absolute expiry date and time after which the delegation rights are automatically removed. Other policy and authorization restrictions and delegation issues should also be addressed like permanence, totality, levels of delegation, etc.
- Some requests inside the *DT_i* will need to be followed by suitable credentials in compliance with the delegate's policy. A formal approach towards such delegation logic has been proposed in [6].
- It should be clearly stated what exactly can be used as a legitimate delegation credential. This will be the Signed Delegation Token (*SDT_i*) as defined above. No other object can be legitimately used for that purpose.

The flexibility in the formation of the *DT_i* makes our protocols ideal for diverse architectures that have been proposed for distributed environments, mobile agent systems, authorization and access control systems and even other frameworks like MExE as it will be shown in section 4.1. However adaptation to specific models is not within the scope of this paper.

3. FLEXIBLE DELEGATION PROTOCOLS

3.1 Multi-Mode Levels of Trust

As previously mentioned, our protocols will adapt themselves to the environment. This flexibility will alleviate the computational burden but in no event will it compromise our basic security requirements which are authentication and accountability of delegation. Our approach depends on the trust of a specific migration that is defined as a function of the device's trust towards the communications medium and the delegate principal.

First we consider the communications medium and we define two different classes C1 and C2.

C1 classification will be adopted when delegation takes place in an environment that is supposed to have its communications links secured. For example this could be realised if one of the following scenarios are applied:

- The terminal connects to a computer with a direct cable or cable connection to a private trusted LAN.
- The terminal enters a trusted PAN or wireless LAN either in a home or office environment. Fundamental security is provided by the underlying technology (Bluetooth, 802.11, etc).

In contrast to this, C2 classification will be adopted when delegation takes place in an environment where communication is not secured and can be intercepted. This could be realised in one of the following scenarios:

- The terminal enters a public ad-hoc network that either offers no security or generally there are concerns of several security risks.
- The terminal uses a (wireless or fixed) access point or delegates through untrusted or unknown networks.

Next we separate delegates in four categories:

a) T1: In this category the mobile device blindly trusts the delegate. For example the delegate could be an agent service operating at a platform like a home PC or a private workstation.
b) T2: The terminal has high levels of trust for the delegate that could be a service or a site that is guaranteed to be indisputable. This could be a service running in a corporate PAN or a ubiquitous service that is certified by the terminal's manufacturer.
c) T3: The terminal has acceptable levels of trust for the delegate. In this case the delegate should bear an acceptable certificate.
d) T4: In this case the terminal does not trust the delegate. Such delegate might for example have a certificate that cannot be authenticated or is verified but has poor or expired attributes.

3.2 Terminal Delegation

All entities that get involved are expected to be or be able to become a part of a generally acceptable network of trust. This could for example be a Public Key Infrastructure (PKI). Every entity should maintain an asymmetric key pair (SK_i, VK_i) that will be registered with this network of trust and will serve the authentication and revocation requirements. We also assume that all entities are capable of dynamically creating new key pairs (DSK_i, DVK_i) in a decentralised and autonomous way and will be totally responsible for keeping safe the signing (private) keys.

Following the previous classification, the terminal delegation will have five different modes of operation to choose from. We define mode *{a}* to be case C1/T1 or case C1/T2; mode *{b}* to be case C1/T3; mode *{c}* to be case C2/T1 or case C2/T2; mode *{d}* to be case C2/T3; mode *{e}* to be case C1/T4 or case C2/T4.

1. First message $\langle A_1 \rightarrow A_2 \rangle$:

$$\{a\} or \langle b \rangle : A_1 \| DT_1$$
$$\{c\} or \{d\} : A_1 \| DT_1 \| r_1' \| SK_{A_1}(A_2 \| DT_1 \| r_1')$$

2. Second message $\langle A_2 \rightarrow A_1 \rangle$:

$\{a\}: DVK_2$
$\{b\} or \{d\}: r_2' \| DVK_2 \| DSK_2(DVK_2) \| SK_{A_2}(A_1 \| DT_1 \| r_2' \| DSK_2(DVK_2))$
$\{c\}: r_2' \| DVK_2 \| SK_{A_2}(A_1 \| DT_1 \| r_2' \| DVK_2)$

3. Final message $\langle A_1 \rightarrow A_2 \rangle$:

$\{a\} or \{b\} or \{c\} or \{d\}: SDT_1$

In mode *{e}* the delegate is an untrusted party and delegation terminates or proceeds according to the user preferences or approved overruled actions.

The delegation in all cases is a protocol involving the exchange of three messages: a delegation request, an acceptance of the request and the actual delegation. The delegate is also required to create a new delegation key pair and send it back for getting its power to delegate authorised. This is achieved by binding the Delegation Token with the delegation key.

In mode *{d}* the request (DT_1) is accompanied with a signature and a timestamp. As a result A_1 is authenticated to A_2 and there is also prevention against reply attack. A_2 in advance creates DVK_2 and SDK_2 and sends back only DVK_2 for A_1 to authorize it. In order to prove the existence of DSK_2 he also uses that key in order to sign DVK_2. A_1 will never be aware of DSK_2 but will still have good reasons to believe that it exists and it is unique. However this knowledge will only improve security when this delegation mechanism is followed on (in a cascade scenario) by other delegation variations. This will be further analysed in section 4.3. A_2 in addition authenticates itself to A_1 by concatenating a signature of this message. The same signature can also be used as evidence by A_1 that A_2 has accepted the specific delegation.

The last part of the protocol is the same for all the modes and consists of A_1 sending A_2 the Signed Delegation Token. As discussed above this is the only token that gives A_2 the power to exercise this delegation.

Next, mode *{c}* is exactly the same with mode *{d}* apart from the delegate's response, which gives no clue about the nature of DVK_2. However, this is not necessary because it is assumed that A_2 does not have any reason for doing so since it belongs to the T1 or T2 domain.

Next we consider mode *{b}* which demands A_2 to send back a signed declaration that he accepts the delegation, in the same way that this happens in mode *{d}*, but now A_1 just sends an unsigned DT_1 in the first message. The reason for the former is that A_2 cannot be sufficiently trusted and the reason for the latter is again efficiency. The hypothesis of a potential man-in-the-middle attack is practically nonexistent as the communication medium can be trusted, for various reasons. One could also claim that in the first message A_1 does not authenticate himself to A_2. As a result A_2 formally accepts a delegation that might originate from a malicious party. However, A_1 is obliged to authenticate itself and authorise A_2 in the final message.

Finally in mode *{a}* we present the most lightweight version of the protocol during which all the cryptographic burden for the terminal is just the creation of a signature in the last message. In this case unilateral authentication is acceptable because A_2 is highly trusted.

3.3 Cascade Delegation

Cascade delegation will also be a three step message exchange. The first two messages will be exactly the same with the terminal delegation according to the new mode of operation. After that the delegate that wants to forward the delegation, will have to bind the new Delegation Verification Key that has been received with the new Delegation Token that has to be legally formed in line with the previous one. Finally the new Signed Delegation Token has to be accompanied by the previous ones. This third message will be the same for every mode and will be the following:

$$\langle A_2 \rightarrow A_3 \rangle : A_1 \| DT_1 \| DVK_2 \| SDT_1 \| SDT_2 \| DSK_2(SDT_2)$$

Cascade delegation can have arbitrary length and in the most general case this message will be the following.

$$\langle A_{i-1} \rightarrow A_i \rangle : C_i(A_{i-2}) \| C_i(DT_{i-2}) \| C_i(DVK_{i-1}) \| C_i(SDT_{i-1}) \| DSK_{i-1}(SDT_{i-1})$$

Please note that in our equations DVK_1 is null data. Also it is not necessary for A_i to verify the whole chain of $C_i(SDT_{i-1})$, except for SDT_{i-1}.

3.4 Delegation at End Point

The last delegate A_f that contacts the end point (B) that provides services has to prove to the latter that he holds a valid SDT_{f-1} and requests a specific service (SRV) to be granted to A_1. The server will have to verify that the service (SRV) complies with the last Delegation Token. In advance, the server will also have to verify that all the Delegation Tokens, which should be attached, comply with their previous ones. This process has to continue iteratively until the whole chain of Delegation Tokens is exhausted and the server finally reaches, verifies and parses DT_1. The message has as follows:

$$\langle A_f \rightarrow B \rangle : C_f(A_f) \| C_f(DT_{f-1}) \| C_f(DVK_f) \| C_f(SDT_{f-1}) \|$$
$$r_f \| SRV \| SK_f(A_f \| B \| SRV \| r_f) \| DSK_f(r_f \| SRV)$$

The delegation at the end point differs from all other steps in the fact that it is suggested to be accomplished with just one message. In terms of accountability, the final delegate is responsible for the proper usage of the SDT_{f-1} that receives from the previous entity as well as the DSK_f that has created. In addition, although A_f binds the SRV with A_1, this does not mean that A_1 is accountable for the SDT_{f-1}. This kind of responsibility is transmitted backwards until the entity that misused the received rules is detected. In the case that all delegates manage to demonstrate legitimate actions, then the chain of responsibility eventually ends in A_1.

The whole delegation ends successfully at the point where B will eventually directly serve the appropriate entity and will send straight back to it all the necessary data, in a secure way. This in most cases will be A_1, but also it can be an arbitrary entity that should be specified in SRV, in compliance with the specifications of DT_1 and the whole chain of $C_f(DT_{f-1})$.

4. ANALYSIS

4.1 Compatibility and Interoperability

The protocol does not require special architectures and it basically depends on the prerequisite of a network of trust (e.g. leveraged by public key identity based certificates). Delegation is decentralised; hence our protocol is scalable and compatible with diverse ubiquitous environments.

The terminal in most cases should operate either on mode *{d}* or *{c}*. These are safe solutions for every case and are thereafter preferable. In specific environments the device will probably be configured. Mode *{b}* could be enough for a trusted PAN. Finally the terminal could be initialised along with a personal mobile agent and probably choose either *{a}* or *{c}*.

Another case involves compatibility with cellular networks or similar networks offering wireless Internet services. At this stage we present compatibility with the MExE specifications [7], although we could also consider other platforms. MExE classifies downloaded material into four distinct domains according to the properties of the accompanying digital signature. For every domain specific permissions are given and the applications' functionality is constrained within a sandbox. Our mechanisms perfectly match with such framework, if we simply consider T2 to be the Operator/Manufacturer domain, T3 to be the trusted third party and T4 to be the untrusted third party. As a result, after initial authentication, the device will be able to adapt to a specific mode of delegation.

4.2 Performance Analysis

In aspects of data transmission, less data has to be exchanged in comparison with other proposals. There are three main reasons for that:

a) Redundant information that a principal is supposed to know about another principal are not resent. As a result even if there are three message exchanges, the total number of bytes is reduced to minimum.
b) Cryptographic data are not used when there is no practical risk for that.
c) The first modes of operation further reduce the messages' length.

In aspects of computational expense, the whole protocol is much more lightweight. The main reasons for that are the following:

a) The usage of cryptographic functions for signing and verifying offers a great advantage over other solutions that suggest heavier asymmetric or symmetric cryptographic computations.
b) The creation of delegation keys, which require expensive battery and CPU consumptions, is done by the delegate rather than the terminal.

Finally the whole model offers a better quality of service as a result of its compact multicasting features. A multicasting scenario can be envisaged when the mobile device enters a PAN that has access to the Internet. It might be desirable to make the same delegation request to a multitude of servers and choose the best one according to their responses. This is impractical with delegation schemes where only one specific agent can be used at a time.

4.3 Security Analysis

Whenever timestamps and/or nonce are used, freshness and uniqueness of the accompanying message is provided. This also prevents against reply and denial-of-service attack. However by just relying on time-stamps we assume that there is relative time synchronization between the two communicating parties. It is envisaged that it is not realistic for mobile terminals to just depend on that assumption. Consequently the usage of nonce is recommended either with or without the presence of a timestamp.

By default the protocol will mainly operate in mode *{d}* or *{c}* where denial-of-service and reply attacks will be prevented by the protocol itself. While operating in mode *{a}* or *{b}* one could argue the fact that either DVK_2 or DT_l are sent completely in plain text. Considering that the terminal will always bind the Delegation Verification Key with its own identity and the identity of A_2, the SDT_l will only have value in the hands of A_2 and only if the genuine DVK_2 has been signed. Although the potential man-in-the middle attack cannot be realised in such an environment, it will still however be easily detected since every entity is always accountable for every action it takes. This can be verified if we make the hypothesis that a fraudulent entity

A_2' manages to impersonate A_2 and further cascade the delegation. At the end point the server will be able to track down the fact that A_2' misused the SDT_1 that was intended for A_2.

Next we need to comment on the difference between mode *{c}* and mode *{d}* which consists of having the delegate explicitly prove that the DVK_i that he sends back for approval is indeed part of an asymmetric key pair. This proof will make the delegate accountable for using DVK_i only as it has been described in section 3.2 and not as a symmetric secret, which is supported by this delegation technique. However if the delegation mechanism is kept uniform then mode *{d}* offers little security improvement over mode *{c}*. On the choice of whether using asymmetric or symmetric delegation keys, it should be mentioned that generally the former should offer more robust accountability due to the fact that the secret key needs not to be shared.

Furthermore, we have to mention that the security of the whole protocol will heavily depend on the security of the public key network of trust and management, which is out of the scope of this paper. This only concerns the authentication part of the protocol, since the delegation is totally decentralised and separate. It is generally assumed that firm regulations govern certificate authorities and/or certificate storage and management.

Another important issue is revocation. This consists of two parts: The first one is revocation of SK_i for which one should refer to public key management background, such as Certification Revocation List (CRL), which is suited for PKI. The second one is revocation of DSK_i, which at this stage is considered negligible as SDT_i should generally be short lived. Hence we strongly recommend the DT_i to have a short expiry date (see Section 2.4). Also, delegation key strength should always be evaluated.

Finally it is recommended for accountability reasons all entities to store in a safe place audit trails for all the signed messages.

5. CONCLUSION

In this paper a first step has been made towards designing ubiquitous delegation algorithms. These algorithms demonstrate robust characteristics like multicasting efficiency and multi-mode operability. We have introduced end-to-end cascaded delegation protocols that demonstrate simplicity and flexibility to adapt to various networks and applications. In addition special care has been taken in order to preserve accountability of delegation.

The described characteristics make this protocol ideal for reconfigurable terminals and ubiquitous services since the proposed cascade delegation activities can take place in the PAN environment or in general purpose hostile networks in a compact way. However, extra work has to be done on detailed public key and trust management issues. In addition it is vital to develop a simulation environment in order to thoroughly evaluate the protocol in different scenarios that will quantify the actual benefit. Continuing this initial work, we expect to provide protocols with greater flexibility and further improve performance. Finding such solutions is considered of great importance for future secure ubiquitous services.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the anonymous referees for their constructive comments and suggestions.

REFERENCES

[1] M. Abadi, M. Burrows, C. Kaufman, and B. Lampson, "Authentication and delegation with smart-cards", Science of Computer Programming, 21:91–113, October 1993.
[2] M. Abadi, M. Burrows, B. Lampson and G. Plotkin, "A calculus for Access Control in Distributed Systems", ACM Transactions on Programming Languages and Systems, Vol. 15, No 4, Pages 706-734, September 1993.
[3] B. Crispo, "Delegation protocols for electronic commerce", Proceedings of the 6th IEEE Symposium on Computers and Communications, July 2001.
[4] Y. Ding, Patrick Horster, Holger Petersen, "A new approach for delegation using hierarchical delegation tokens", Proc. 2nd Conference on Computer and Multimedia Security, Chapman and Hall, 1996, S. 128-143.
[5] M. Gasser, A. Goldstein, C. Kaufman, and B. Lampson, "The digital distributed system security architecture", Proceedings of the National Computer Security Conference, 1989.
[6] N. Li, B. N. Grosof, and J. Feigenbaum, "Delegation Logic: A logic-based approach to distributed authorization", ACM Transactions on Information and System Security (TISSEC), in press, February 2003.
[7] Mobile Execution Environment Forum, "MExE specifications", Available online from http://www.mexeforum.org
[8] K. R. Sollins, "Cascaded authentication", In Proceedings of the 1988 IEEE Symposium on Security and Privacy, pages 156–163, April 1988.
[9] V. Varadharajan, P. Allen, and S. Black, "An analysis of the proxy problem in distributed systems", In IEEE Symposium on Security and Privacy, pages 255–277, 1991.
[10] C. Y. Yeun, and T.Farnham, "Secure Software Download for Programmable Mobile User Equipment", IEE 3G Mobile Communication Technologies Conference, pages 505-510, 8-10 May 2002.
[11] C. Y. Yeun, G. Kalogridis, and G. Clemo, "Secure Mobile Delegation for Future Reconfigurable Terminals and Applications", In Proceedings of the 1st Software Defined Radio Technical Conference, November 2002.