Digital Rights Management in Ubiquitous Computing

Madjid Merabti and David Llewellyn-Jones
Liverpool John Moores University

The Internet has revolutionized multimedia content distribution, shifting the way content producers and users approach digital rights. However, ubiquitous computing will alter the digital rights management environment even more, and current techniques are ill equipped to deal with the changes. Distributed trust can help to overcome the challenge of maintaining digital rights for ubiquitous computing, using cellular automata to measure trust levels across a system.

Ubiquitous access to computers and networking has revolutionized the availability and distribution of digital multimedia content. For content producers, the ease and diversity of methods by which they can disseminate their output has significantly increased over the last two decades. For content users, the wider dissemination of multimedia content has resulted in higher quality and more choices.

Many of these developments have been driven by the Internet's success. It's easier than ever for people to find the content they want by typing a few search terms into Google or a peer-to-peer file-sharing client. Extending beyond the desktop's boundaries, small, portable, and wirelessly networked devices such as phones and PDAs are becoming increasingly multimedia capable and represent important markets for content.

The ease with which users and creators can share multimedia content and the proliferation and increasing variety of devices that can handle this content produces obvious benefits. But it also brings drawbacks, forcing traditional content producers to face difficult questions. Most fundamentally, in a world of pervasive data sharing, how can content providers ensure that they're suitably recompensed and that their copyright is respected? This challenge is only set to become more difficult as we move closer to ubiquitous computing.

We propose using trust as a way to ensure that content providers' rights are respected, while allowing a suitable flow of content. Broadly speaking, the aim is to assign levels of trust to nodes within a network. As content moves through the network, we can establish node responsibility with a system based on cellular automata techniques in combination with techniques to determine when digital rights infringement occurs and when users are acting legitimately. Hence, we could let content and data flow relatively unimpeded where trust levels suggest that users are acting legitimately. In areas where frequent infringement occurs, we could impose more restrictions and digital rights management (DRM) techniques.

The techniques we're proposing are novel and still under development, so it would be grossly premature to suggest we can provide a solution to all the difficulties involved. Nevertheless, this is an area that multimedia research is bound to tackle, and at the very least, we hope to show that we can apply new perspectives to the problem.

Ubiquitous computing

Ubiquitous computing's vision represents the inevitable fulfillment of the trends in hardware, software, networking, and social attitudes that we can see clearly in computing today. Yet, despite the fact that hardware capabilities have now surpassed those found in Mark Weiser's vision (see http://www.ubiq.com/hypertext/weiser/UbiHome.html), ubiquitous computing's full potential and its impact on the way people access content has yet to be fulfilled.[1] Despite wide-scale use of multimedia content on mobile and networked devices, the industry has only partially achieved the anticipated move toward the fluid flow of data between devices. Data and software are still tied to particular devices with networks tightly segregated to ensure security. For example, the ideal of home networking with smart appliances collaborating for the user's benefit is hampered by the expert knowledge needed to maintain such a system.

Nonetheless, the evolution of computing suggests a continued gravitation toward the full ubiquitous computing vision, and research and development continue to aim toward this goal. The user's present focus remains directed at individual hardware devices when it comes to computer use. In contrast, ubiquitous computing shifts away from hardware so that individuals focus on data ownership rather than hardware.[2,3] When we talk about *disappearing hardware*, we're not just considering devices without a user interface but, in fact, something more intrinsic. The user should be hardware agnostic and care only about the data they can access.

1070-986X/06/$20.00 © 2006 IEEE

Current DRM-Enabling Technologies

Current efforts at protecting digital content have focused on technologies that work in isolation on an individual machine. The following technologies represent a number of such developments aimed at either reducing the opportunity to copy copyrighted material or detecting such material when it has been copied.

Trusted Computing Platform

At the core of the Trusted Computing Platform[1,2] is the inclusion of a tamper-proof hardware component built into all protected computers and devices. This component acts in much the same way as a smart card, providing cryptographic functions that the device vendor can assume to be reliable by virtue of the component's tamper-proof nature.

Crucially from a Digital Rights Management (DRM) perspective, the Trusted Computing Platform allows trustworthy and reliable configuration reporting from the device vendor's perspective, acting as the content suppliers' agent.

Digital watermarking

A digital watermark[3,4] constitutes additional information that's hidden within media data (such as audio or video) so that the watermark can be recognized computationally but is imperceptible to a human eye or ear. A media player will recognize the watermark so that it can positively identify the material being played and thereby ensure that the user has a legitimate right to access it.

To be effective, the watermark must be robust against transformations of the media data. For example, it should survive even if a video file is converted to an analog format and then back to a digital format.

Fingerprinting

Media data can be characterized by its raw signal representation, without the need to include additional identification data, such as the ID3 tags present in many MPEG-1 Audio Layer 3 (mp3) files or watermarking techniques. Such a characterization is known as the media's fingerprint.[5,6] Among other purposes, we can use fingerprinting for DRM because it lets us identify content independently of its format. Thus, a file can be identified and checked against a database of files that the user is entitled to access.

A second form of fingerprint technology can be found in the DRM literature. Content producers are conscious of the fact that deviant users will aim to remove content metadata, signature, or watermark protection to bypass security and then distribute content illegitimately. Hence, they've developed techniques that leave behind evidence identifying users if they attempt to break the DRM security.[7,8] A simple example is a watermark that contains redundant data unique to the content sent to a particular user. Attempts to remove the watermark might leave behind this unique data that identifies the copy's original owner.

References

1. S. Pearson et al., *Trusted Computing Platforms: TCPA Technology In Context*, Prentice Hall, 2003.
2. "TCG Specification Architecture Overview, Revision 1.2," Trusted Computing Group, 2004.
3. F.A.P. Petitcolas and R.J. Anderson, "Evaluation of Copyright Marking Systems," *Proc. Int'l Conf. Multimedia Computing and Systems*, vol. 1, IEEE CS Press, 1999, pp. 574-579.
4. D. Kirovski, H. Malvar, and Y. Yacobi, "A Dual Watermark-Fingerprint System," *IEEE MultiMedia*, vol. 11, no. 3, 2004, pp. 59-73.
5. M. Clausen and F. Kurth, "A Unified Approach to Content-Based and Fault-Tolerant Music Recognition," *IEEE Trans. Multimedia*, vol. 6, no. 5, 2004, pp. 717-731.
6. E. Allamanche et al., "Content Based Identification of Audio Material Using MPEG-7 Low Level Description," *Proc. Int'l Symp. Music Information Retrieval*, 2001; http://ismir2001.ismir.net/.
7. M. Fernandez and M. Soriano, "Soft-Decision Tracing in Fingerprinted Multimedia Content," *IEEE MultiMedia*, vol. 11, no. 2, 2004, pp. 38-48.
8. R. Safavi-Naini and Y. Wang, "Sequential Traitor Tracing," *IEEE Trans. Information Theory*, vol. 49, no. 5, May 2003, pp. 1319-1326.

Digital rights management

Researchers have developed numerous DRM techniques to overcome the problems introduced with widespread digital content distribution over the Internet. Such techniques are aimed at ensuring that content providers can benefit from their work and that their copyright and digital rights are respected. (The "Current DRM-Enabling Technologies" sidebar details several of these techniques.) However, the notion of disappearing hardware causes significant problems for most of the solutions currently in development. Such methods can't accommodate the flexibility of content movement that ubiquitous computing demands.

For example, consider a possible scenario: If a user is reading an article on a desktop computer, she'll expect the article to also be available on a handheld device. In addition, she'll want the device to activate simply because she picked it up, or moved away from her desk, even if she doesn't own the device in question. She might

also want the article displayed at the same position in the document. This seamless user experience is harder to achieve with every restriction placed on data movement through a network.

Clearly, this liberation of data will also have a profound effect on the ways we can achieve DRM in a ubiquitous computing environment. First, it exacerbates all the problems and challenges the DRM field currently faces. As we've seen, the multitude of devices forming a ubiquitous computing environment are all highly networked, allowing fluid data transfer between them. The proliferation of devices will be achieved largely through an increase in lower-power and reduced-resource devices. This hinders the liberal application of heavy-duty data-encryption techniques for data protection.

It also brings with it new difficulties. A more flexible attitude to hardware ownership means that there might not be a single owner for any particular hardware device. Conversely, users will expect easy access to their data from multiple devices, often simultaneously. Common content controls, such as those built on the Trusted Computing Platform,[4] often force access restrictions to a single device or to media in a single format, possibly preventing it from being transferred from one physical medium to another. Such restrictions aren't compatible with ubiquitous computing's goals.[5,6] Moreover, these techniques don't generalize or scale in the ways ubiquitous computing demands. A new perspective is necessary.

DRM is by no means unique in this respect, and we find precedent in other areas such as security, where ubiquitous computing precipitated the loss of the *perimeter model* (where, for example, firewalls and intrusion detection systems form a boundary around a safe, but otherwise vulnerable, soft center), thus demanding the innovation of alternative techniques.

Using trust to protect data

The three major aspects of current DRM techniques that preclude their use in a ubiquitous setting are their

- inflexibility with regard to fluid data flow,
- centralized nature that relies on enforcement from a central point—usually the content producer or its agent, and
- considerable computing power requirements, for example, for watermark checking.

In contrast, using distributed trust to protect content producers' digital rights means the community enforces the application of digital rights. Each device in the community has a part in enforcing the digital rights of the data within the network as well as a stake in maintaining the process' integrity. In this way, we remove any central weak point while distributing the effort required among numerous devices.

Undoubtedly, we could achieve a distributed trust paradigm such as this in many ways, and we hope future work in the area will develop the concept further and in new directions. We present an initial possibility based on our own work that illustrates the viability of using distributed trust to manage digital rights.

In addition to the benefits we've already cited, because of our technique's distributed nature, we can also allow limited content distribution falling under the terms of fair use. Rather than relying on enforcement from a single centralized point, the community enforces compliance by ostracizing deviant users from the community, preventing them from gaining access to further content.

Our system, based on cellular automata techniques, works by curtailing the distribution of copyrighted material or software. We add a thin trust layer to all devices that will let a device's neighbors make judgments about its trustworthiness. The process relies on two key characteristics. First, integrity is maintained through information sharing. This means that all of a device's neighbors will have an influence on its claimed trustworthiness. This reduces the possibility of any individual device or group of devices subverting the system. Second, although data is shared directly only with a device's neighbors, we can harness the emergent properties of the cellular automata network to ensure that consequences have a wider effect.

Building communities

There are numerous ways to deploy trust-based systems such as the one we describe here. Similar to other manually maintained trust mechanisms (such as the buyer and seller feedback implementation used by online sites like ebay[7] or PGP's trust and sign mechanism[8]), our work's focus has been on creating communities within the ubiquitous computing environment. Our communities are populated by stakeholders who benefit from being members of the community and from maintaining trust within that community to police digital rights.

There are two potential ways to maintain collaboration in such a situation. We can maintain it either with the incentive of mutual reward and benefit that arises from community membership or through fear of the retribution that might result from circumnavigating necessary checks or processes.

For example, a company might offer consumers willing to participate in the framework the opportunity to access improved, less restricted, or cheaper content. One of the benefits to members of such a community would be greater freedom to use the material. From the company's point of view, anyone consistently flouting copyright requirements would be penalized.

A trust mechanism's distributed nature also makes it ideal for more ad-hoc, unmanaged communities where content doesn't necessarily emanate from a single source. The Creative Commons movement[9] is an example of such a community aimed at content production and provision on a wider scale, with a well-defined but flexible and more holistic attitude toward copyright.

The scheme we present doesn't require an assumption of universal collaboration for it to work. However, we assume a suitable level of benefit or retribution to enable a sufficient number of devices to collaborate effectively within the framework.

Trust architecture

The cellular automata trust layer we propose sits between the network stack and the normal application layer. All data arriving across the network passes through the trust layer. Some of this data, depending on the state of the cellular automaton cell at the node, is checked using standard DRM methods—for example, checking media files' watermark integrity or fingerprints. Due to the processing requirements for watermark checking (or equivalent), we won't want to check all data. Again, depending on the current state, a portion of the data is sent directly on to the application for it to deal with. This latter data therefore passes from the network stack to the rest of the operating system just as it would under normal circumstances.

Figure 1 shows this process and compares it to normal network functionality. The watermark checking's result is fed back into the trust layer so it can react to any illegitimate data discovered.

To work effectively, the trust layer and watermark checking must have some knowledge of the structure and type of data it's dealing with. However, the best way to implement this is outside this article's scope.

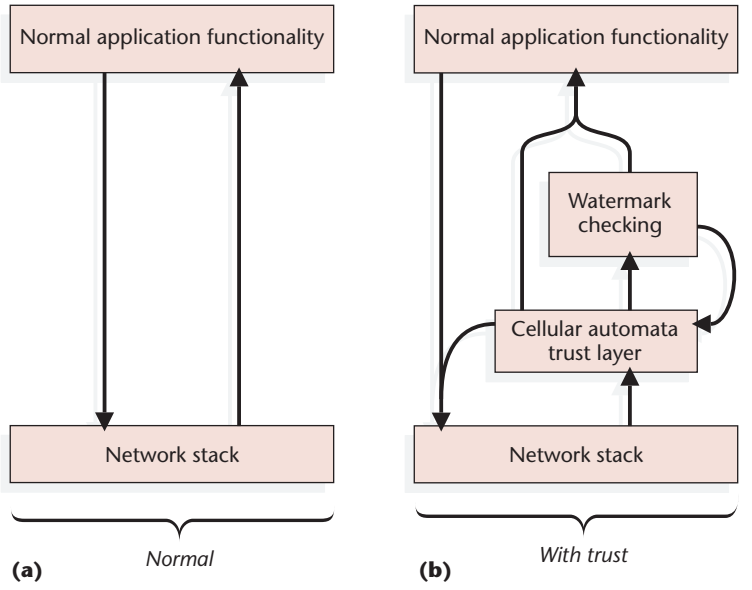


Figure 1. To protect digital rights in a ubiquitous environment, we can modify (a) a traditional network with (b) a cellular automata trust layer to form a trust architecture.

The trust layer has some network overhead itself, but it's negligible. (We go into more detail on this point later.)

Because we aren't considering the watermark checking techniques in this article, everything we're interested in occurs within the cellular automata trust layer. The cellular automata helps us keep this layer exceptionally thin to avoid unnecessary overhead.

In effect, all this layer does is periodically execute a small mathematical function. The function uses the result returned by the watermark checking as parameters, along with data received over the network from the device's immediate neighbors.

Using this function, we determine

- whether the current data should be checked for legitimacy and
- the data that must be sent to surrounding nodes to maintain the system's security.

The processor and network requirements for the cellular automata to operate are incredibly low. To see this, we must consider the transition function we use.

Cellular-automata-based system

The transition function is a simple mathematical function that we can easily code that

Cellular Automata

Cellular automata, which John von Neumann introduced in the late 1940s,[1] characteristically involve a regular grid of cells, each of which has an individual state. A discrete time dimension is added, and at each temporal step, a transition function is applied that takes as inputs the current state of the cell and its neighbors and that outputs the cell's new state. The transition function is applied at each step and for every cell simultaneously. (A large body of literature exists describing properties of cellular automata, for example, by Stephen Wolfram[2] and many others, including mainstream publications.[3])

The most well-known example of a cellular automaton is John Conway's "Game of Life,"[4] introduced in the 1970s. This takes a grid of squares where each cell is either dead or alive.

At each time step, a cell's life status will change depending on how many living cells surround it. Figure A shows the three simple rules. However, the resulting behavior on a large grid is highly complex, so complex in fact that it has been shown to be Turing complete. This complexity underlines cellular automata's beauty and utility—by applying simple rules to each node, highly complex emergent behavior can result.

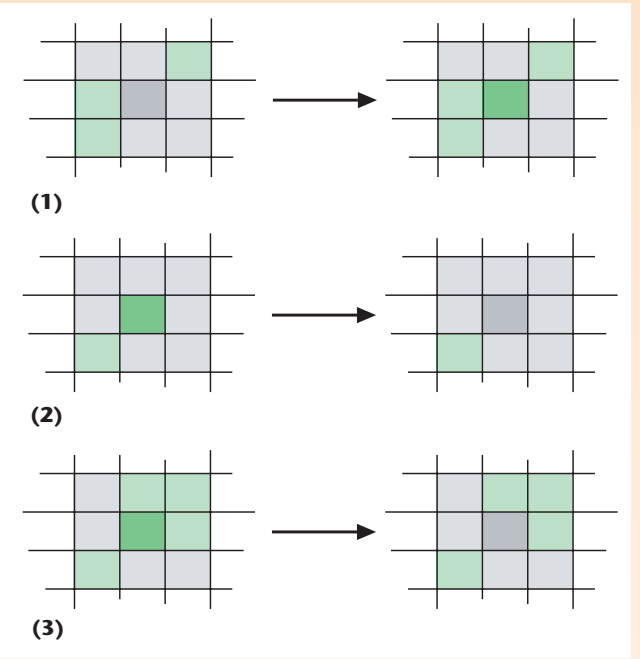


Figure A. Three rules of a cellular automaton in Conway's "Game of Life:" (1) A dead cell surrounded by exactly three live cells will become alive. (2) A live cell surrounded by one or fewer live cells gets lonely and will die. (3) A live cell surrounded by four or more live cells gets claustrophobic and will die.

References

1. J. von Neumann, "Theory of Self-Reproducing Automata," *University of Illinois Lectures on the Theory and Organization of Complicated Automata*, A.W. Burkes, ed., Univ. of Illinois Press, 1949.
2. S. Wolfram, "Statistical Mechanics of Cellular Automata," *Reviews of Modern Physics*, vol. 55, July 1983, pp. 601-644.
3. J. Vlietinck, "2D Waves," *Archimedes World*, May 1993, pp. 16-17.
4. M. Gardner, "Mathematical Games: The Fantastic Combinations of John Conway's New Solitaire Game 'Life,'" *Scientific Am.*, Oct.1970, pp. 120-123.

requires little in the way of processing power. It forms the basis of the technique that lets us consider the network devices as cellular automata nodes. Cellular automata are well-known mathematical constructs, although they aren't fully understood. (See the "Cellular Automata" sidebar for more information.)

The crucial aspect that allows cellular automata to be useful for our purposes is their reliance on many devices sharing information locally to create complex properties that emerge across an entire network of devices. The devices (or cells) are usually arranged in a grid formation, and each cell maintains its own state. As time progresses, a cell will alter its state deterministically based on local information about the state of its immediate neighbors. A cell need not have knowledge about its current state or the wider network, and the transition function that a cell uses to update its state can be simple. Yet, the consequent behavior across the entire network can be highly complex, and small changes that occur in one part of the network can have global effects.

The transition function will take in a node's current state and the state of its surrounding nodes and output the current node's new state. This function would normally be used uniformly across a grid of cells, which is how we will use it. However, unlike a standard cellular automata, we arrange our cells in a more complex network.

The transition function acts on the current state at a node over discrete time steps. To define our transition function, let's take t to be a discrete time variable and $s_n(t)$ to be the state at time t for node n. The job of the transition function f is to determine the value of $s_n(t + 1)$. Thus, $s_n(t + 1) = f(s_n(t))$. To define this simple transition function, we use the following functions:

$$l_n(t + 1) = l_n(t) + v_n(t + 1) \quad (1)$$

$$v_n(t+1) = \left(v_n(t) + \frac{1}{M} \left(G \cdot M + \frac{T}{k} \sum_{0<i\le k} \left(l_i(t) - l_n(t) \right) + \chi_n^A(t+1) \cdot F_S + \tau_n(t) \cdot F_T \right) \right) \quad (2)$$

and

$$\tau_n(t+1) = \min\left(\max\left(\tau_n(t) - \chi_n^D(t) \cdot T_D + \left(1 - \chi_n^D(t)\right) \cdot T_I, T_M \right), 0 \right) \quad (3)$$

To fully understand these equations, we need to define the constants and additional functions χ_n^A and χ_n^D. We will do this later on, but first we can explain how we use these functions. The state $s_n(t)$ at time t consists of three rational (floating-point) values l_n, v_n, and τ_n—or a node's *legitimacy*, *velocity*, and *trust*, respectively. Velocity is simply the rate of change of legitimacy over time. Therefore, we can represent s_n as the vector $(l_n, v_n, \tau_n)^T$. The transition function f is defined using the three iterative functions as follows:

$$f\left(s_n(t)\right) = f\begin{pmatrix} l_n(t) \\ v_n(t) \\ \tau_n(t) \end{pmatrix} = \begin{pmatrix} l_n(t+1) \\ v_n(t+1) \\ \tau_n(t+1) \end{pmatrix} \quad (4)$$

As we can see from Equation 2, the important pieces of information needed to calculate f are the values $l_i(t)$ for $0 < i \le k$. These are the values $l_i(t)$ for the k neighboring nodes of node n. We can represent each of these state values as a single-floating variable, which we transmit across the network. Hence, little data needs to be shared between nodes, and this data travels only short distances, between neighboring nodes. However, the states need to be updated frequently, which we discuss in more detail later.

To complete the picture, we must consider the step functions χ_n^A and χ_n^D. The χ_n^A function represents whether the node should be checking the current data (whether the node is active) or not. If χ_n^D takes the value 1, then data should be checked, otherwise it can be sent directly on to the application as normal. The function is determined as follows:

$$\chi_n^A(t+1) = \begin{cases} 1 & \text{if } l_n < 0, \\ 0 & \text{if } l_n > L_S, \\ \chi_n^A(t) & \text{otherwise} \end{cases} \quad (5)$$

Table 1. Constants we used in our experiments.

Constant	Approximate Value	Description
F_S	0.88	Security force
L_S	0.1	Security inactive level
M	1.0	Node mass
G	$-F_s \times p$	Gravity*
T	0.3	Tension
R	0.9	Resistance
F_T	F_s + G	Trust force
T_I	0.1	Trust increase
T_D	10.0	Trust decrease
T_M	–20.0	Trust minimum

* This value p ∈ (0, 1) represents the approximate time each node is active.

As we can see, $l_n(t)$ determines the state of χ_n^A. If $l_n(t)$ falls below 0, the node becomes active. If it rises above L_S, it becomes inactive. Otherwise, it remains in its current state.

The function χ_n^D shows whether illegitimate data has been detected. In other words, it's the value returned from the watermark or other checking process. In this case, we represent it as

$$\chi_n^D(t) = \begin{cases} 1 & \text{if a node } n \text{ detects illegitimate} \\ & \quad \text{data arriving,} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

(Note that a node can only detect illegitimate data while in an active state.)

The functions' behavior is strongly influenced by the constants we use, and these constants will depend on circumstantial aspects such as the nature of the network and how reactive the system should be. To give a rough idea about the values these constants might take, see Table 1, which shows the values we used in our experiments.

Interpreting the transition function

The specific details of the transition function give an idea of how the process works in terms of implementation, but they don't give any indication as to what the middleware layer actually does. To give a better understanding, we consider our prototype implementation. Imagine that we've arranged the nodes in a network in a grid. (This doesn't represent a realistic scenario, but it's important for gaining a better understanding of the process.) Figure 2a (next page) shows such an arrangement with the nodes represented as squares on a rectilinear grid, and Figure 2b shows the same nodes as points.

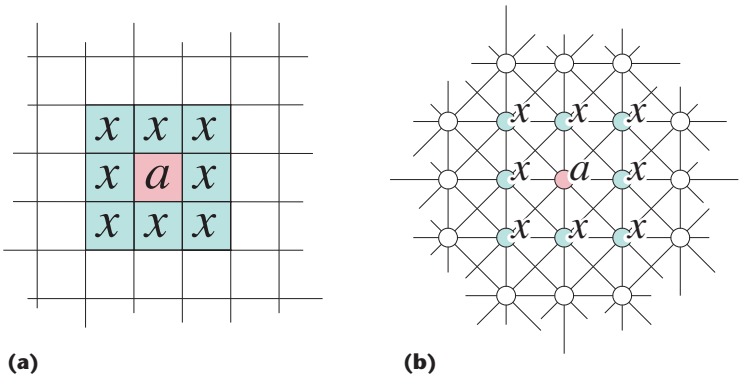


Figure 2. Considering a network as a cellular automaton with (a) nodes represented as squares on a rectilinear grid and (b) the same nodes as points in a network.

For a rectilinear network, at each time step, a node will send a small amount of data—the value of $l_n(t)$ calculated using Equations 1 through 6 — to each of its four neighbors. In a more complex network, there might be more or less neighbors than this. The nodes receiving these values will run all of them through the transition function to establish their new values for $l_n(t + 1)$. The important aspect of the transition function isn't its explicit details but rather the emergent properties that result from it. In this case, we can interpret the resultant behavior to be like an elastic blanket.

So, suppose node n transmits some illegally copied data to its neighboring node m. Once the data is recognized by node m, the node will react by decreasing the trust value associated with the node n. The trust value represents how trustworthy the neighboring nodes consider a node to be. In turn, the reduced trust value will cause the node's legitimacy value to decrease and become negative, meaning the node isn't reputable.

An important distinction exists between trust and legitimacy. A node might appear untrustworthy without it actually doing anything wrong. However, if the community agrees that a node can't be trusted, the consequence is that it loses legitimacy, while a single incident to a single neighbor (which might turn out to be a false accusation) wouldn't matter. Community decision making is thus built into the transition function.

Suppose we represent a node's legitimacy value by its height. Then under normal circumstances the network might look something like the image in Figure 3a. If a node starts distributing illegally copied material, however, the result will become more like the image in Figure 3b.

As Figure 3 shows, the surrounding nodes' negative trust value causes the legitimacy values to reduce, which has a ripple-like effect further out, causing a dip in the legitimacy values of all nodes radiating outward. The consequences of this are three-fold. The reduced values of l_n for each of the surrounding nodes will cause χ_n^A in Equation 5 to remain fixed with value 1. Hence, the surrounding nodes that were occasionally checking content subsequently become far more diligent, checking all the content from n.

Crucially, this provides an additional incentive for the neighboring nodes to take countermeasures against the rogue node n because their legitimacy is affected adversely by its activities. This legitimacy is maintained externally by all the surrounding nodes. If a node doesn't act against the rogue node, it's effectively seen as being complicit. Because the process is massively distributed, every node in the network plays an equal role, and hence, limited opportunity exists for a node to subvert the process for its own ends.

Finally, the rogue node and any complicit nodes will become completely isolated. A still shot taken from our experiments (see Figure 4) shows this clearly. The activated nodes surround the rogue (magenta) nodes and isolate them from the rest of the network.

Isolating nodes in this way, unfortunately, presents an opportunity for denial of service (DoS) attacks. Avoiding such attacks will rely on the careful balance between allowing some small

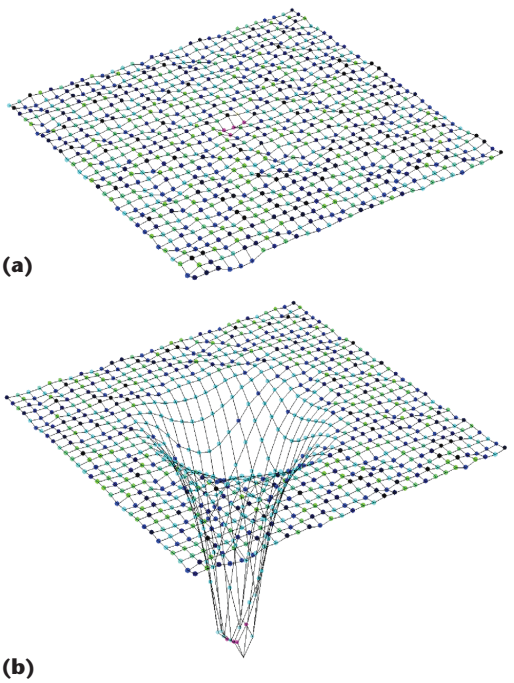


Figure 3. The network projected into 3-space with node legitimacy represented as height, showing (a) normal network behaviors and (b) the effect of a rogue node.

amount of illegitimate data distribution and reacting aggressively to its discovery. We believe that the distributed and communal nature of our proposed scheme will make DoS attacks less likely because numerous nodes would have to work together to launch an effective DoS attack. Nonetheless, this area requires further work using prototypes.

Reducing resource requirements, increasing robustness

Using a simulated Klemm-Eguíluz-generated network[10] consisting of 10,000 nodes, we tested the effectiveness of our method to establish if we could usefully isolate rogue nodes.

Our results were positive and suggest that this would be an effective, scalable way to enforce DRM in ubiquitous computing. In our experiments, we were especially interested in discovering whether we could use such a system in a resource-limited environment.

There are two significant hurdles to overcome when considering DRM enforcement for ubiquitous computing. First, the resources required to identify illegitimate content might be significant—for example, with watermark testing. Therefore, we can't assume that a device can test every single piece of content or data it receives. Thus, the system must work effectively even if only a limited amount of data is tested. Second, we can't be sure that every device will participate correctly. Some devices might not have the resources to implement any security, while others might choose not to participate to cause problems. The system must therefore be robust in the face of potentially inactive or rogue nodes.

We performed experiments in a simulated network, where the movement of illegitimate data was tracked around the network. We chose some nodes to act responsibly with others willfully distributing illegitimate data. Our intent was to test the improvements gained by using our cellular automata technique over a standard checking method. In the standard case, DRM checking occurred periodically on the same proportion of data as our cellular automata method, so that it required the same processing overhead. However, in the standard case, each node acts in isolation, with no collaboration or information sharing between nodes.

In each experiment we did, we set only a fixed proportion of the nodes to participate in the DRM checking. We set this proportion of willing nodes to 85 percent so that 15 percent of the nodes on the network took no part in either DRM checking scheme.

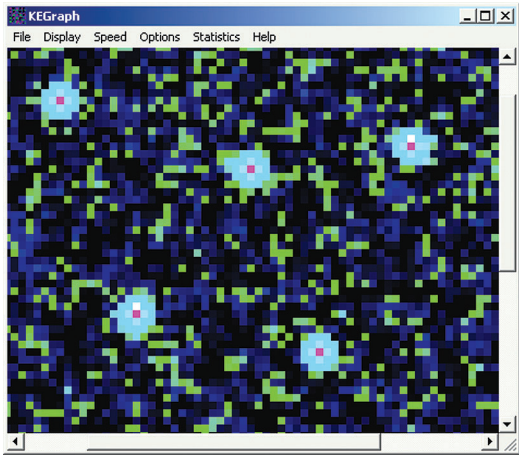


Figure 4. Rogue nodes (magenta) isolated from the rest of the network's active nodes (green/cyan).

For the 85 percent of nodes that were undertaking analysis, we varied the percentage of data being checked at each node between 20 and 80 percent in one-percent increments. For each of these 60 percentage levels, we ran two simulations, one using the cellular automata technique and one using the standard technique, to establish the effectiveness of the two methods. In this way, we aimed to establish the extent to which reducing the resources used for checking affected a rogue node's ability to distribute illegitimate data. Figure 5 (next page) shows the results we obtained.

Figure 5 shows the stark contrast between the standard and trust-enabled models. First, in all cases, the trust-enabled model performed better, often significantly so. Second, the trust-enabled model remains far more effective even when the time spent actively checking data greatly decreases. The consistency with which the trust-enabled network can prevent the dissemination of illegitimate material—even when checking infrequently—is surprising. In Figure 6, we can see this more clearly by taking a slice out of each 3D graph in Figure 5 at the 20-percent time active point.

Looking at Figure 6, we see the extent of illegitimate content distribution using the two methods. In the standard case, approximately 70 percent of the nodes distributed illegal content. With the trust-enabled nodes, illegal content sharing was restricted to just more than 15 percent of nodes. Moreover, the untrustworthy nodes became entirely isolated and could quickly be identified, as we saw in Figure 4.

Simulations using alternative parameters—differing levels of content transfer rates and differing proportions of enabled nodes—produced similar results.

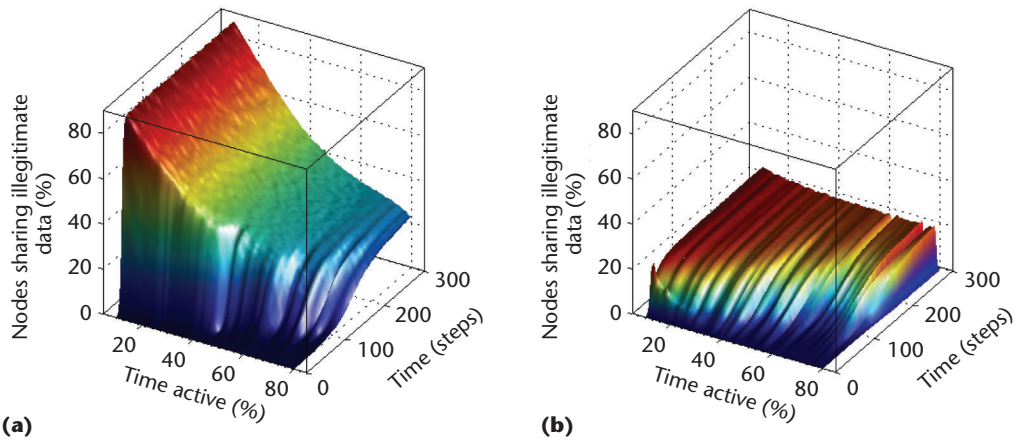


Figure 5. Simulated results with 85 percent of the nodes enabled using (a) standard methods and (b) the cellular automata method.

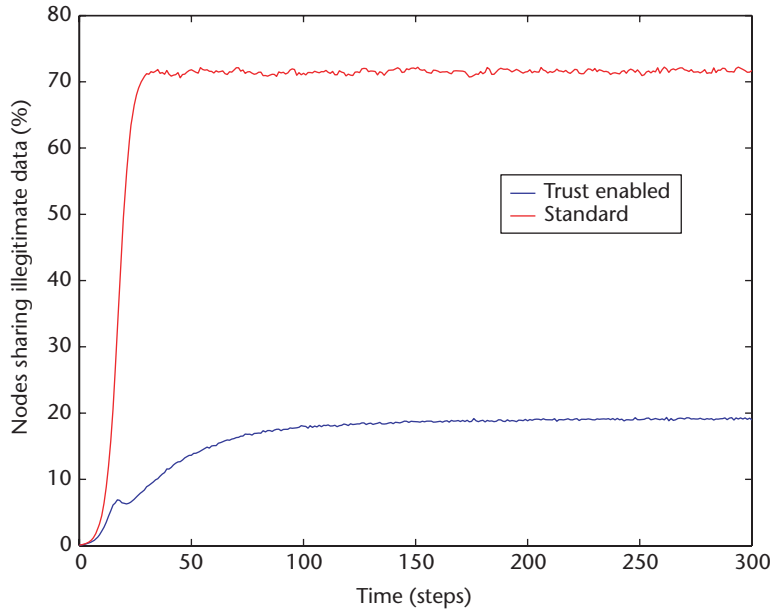


Figure 6. Simulation results comparing illegitimate data distribution between the standard and cellular automata methods with 85 percent of the nodes enabled and 20 percent active.

We should mention a number of caveats in relation to these tests. The Klemm-Eguíluz network has small-world and power-law properties that many network structures are known to exhibit, including the Web, the Internet at the autonomous systems level, and many social networks. In our case, we can view such networks as a community network or a network of ubiquitous computing devices. In the former case, neighbors are logical community neighbors, rather than physical neighbors. The system must dynamically maintain links based on which devices communicate with each other. We didn't include such dynamic links as part of our simulations.

In the latter case, each device would be expected to run the trust-based layer. Because we'd expect routing to occur in a potentially ad-hoc nature within the ubiquitous computing environment, it would be necessary for devices to be able to read and potentially check data (depending on the trust status we described earlier) as it passes through the device en route to another device. In this way, a device's neighbors can be maintained while still using the routing abstraction inherent in IP network addressing.

Real-world implementation

Our current work is directed toward creating an implementation that we can test on a real (rather than simulated) network. Based on the architecture we've described, this is initially a stand-alone application for use within a networked appliance prototype.[11] Ultimately, however, we intend to integrate it into a peer-to-peer file-sharing application that can be deployed more widely outside of our isolated laboratory testing network.

In the longer term, the question remains as to how to persuade people to use such a system in the real world. After all, little immediate and natural incentive exists for users to want to participate in a scheme that restricts what they can and can't do.

Several traits are advantageous to such a system in this respect. Primarily we intend the incentive to come from the community itself so that, for example, inclusion in the community through use of the system will reward the users. However, technical advantages of the system also help.

Crucially, the distributed system is intended to work even when not all nodes are actively participating (see Figure 7). Figure 7 shows the saturation level of nodes sharing illegitimate data, with the proportion of enabled nodes (nodes par-

ticipating in the scheme) varying between 50 and 100 percent. The saturation level represents the maximum proportion of nodes that can freely transmit illegitimate data between each other due to "holes" that arise through the patchiness in participation coverage. The graph shows the situation when nodes are set to be active only 25 percent of the time—a relatively low amount. For example, with 50 percent of all nodes enabled, each checking 100 percent of data received, the worst possible situation is that the remaining 50 percent can freely transmit illegitimate data between them.

Figure 7 also shows that the trust-enabled method consistently outperforms the standard method. More importantly, however, is that with over two thirds of nodes participating, even when only active 25 percent of the time, the trust-enabled model had a positive impact on identifying and preventing illegitimate data transfer.

We can conclude that the trust-enabled process might be useful even with less than full participation. However, the specific results in Figure 7 should be taken with some caution because they apply only to the specific parameters tested using the simulation. As the network characteristics change, the point at which the process becomes effective is also likely to change.

Looking at real-world scenarios, our assertion of effectiveness is reinforced because, in general, only a few users would circumvent any trust-based process such as the one we present here. A recent survey of Internet users in the US by the Pew Internet and American Life Project[12] concluded that of the 22 percent of Internet users who download music, 21 percent say they get video or music files via peer-to-peer services (amounting to only around 5 percent of Internet users).

For the users who aren't distributing illegitimate data, the benefits of using the system overall are based on the increased flexibility that the system affords, the reduced resources required on each individual device, and benefits that derive from the community's existence.

It's also an important feature of the system that checking occurs at the receiver's end. Thus, the recognition of a device distributing illegitimate data occurs on the surrounding devices, rather than the device undertaking the distribution. This acts as a disincentive for users hoping to circumvent the system because they wouldn't be able to safely send data to another user unless they could be certain that the user was also circumventing it.

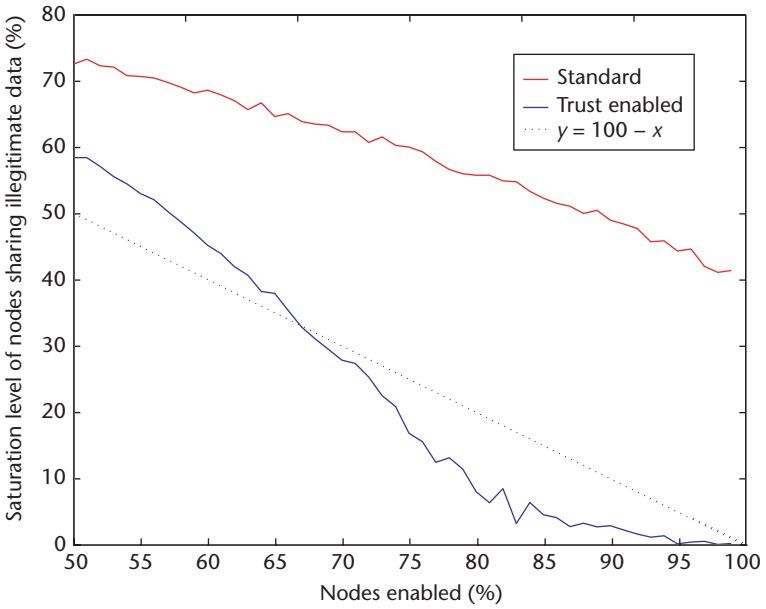


Figure 7. Simulation results of the standard and trust-enabled systems. Fifty to 100 percent of nodes are enabled, while 25 percent are active. The dotted diagonal line represents the hypothetical worst-case scenario, in the case when nodes are active 100 percent of the time.

Conclusions

Effective DRM in a ubiquitous computing environment will rely on finding a technological, legal, and social middle ground between the needs of content providers, users, and technology. Current technical solutions are a long way from this middle ground, either because they're too weak to provide suitable protection, too draconian for users to accept them, or too resource intensive to be used on mobile or embedded devices.

We believe distributed trust might provide a step toward this balance. It won't replace current DRM techniques, such as watermarking and fingerprinting, but rather integrate with them to make them more effective.

Its real strength is that it relies on the collaborative sharing of data between devices, a technique that has been successfully applied to other areas such as spam filtering.[13] By sharing information about devices' trustworthiness, it lets us implement current techniques far more effectively. Moreover, by harnessing a network's emergent properties, we find that little data needs to be shared, and this data need only be shared between direct neighboring devices to enable effective protection across the network.

Nonetheless, the techniques we describe are still at a relatively early stage of development. Simulation results are encouraging, but we must establish the system's effectiveness in real-world scenarios. In addition, we must establish communities that provide a suitable balance between

general incentives and the specific benefits obtained by circumnavigating trust systems such as this one. This balance is crucial in ensuring that enough users participate in order to maintain the system's effectiveness.

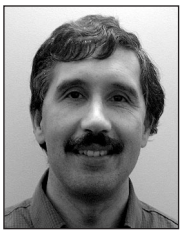
Our research aim isn't to address this issue. Yet, it's an important question, the answer to which might only become clear once we've overcome the technological difficulties of deploying a wide-scale solution.

Although we've applied the techniques we described here to DRM, they might ultimately prove beneficial to other areas such as virus protection and privacy technologies. In ubiquitous computing, all these areas present significant hurdles, though none are likely to be as tricky as those presented by DRM's conflicting requirements.

We believe we've provided a foundation for much further work in the area of distributed trust for DRM. We're currently preparing a prototype with the aim of reaffirming our promising simulated results across a real-world network. Ideally, we'd like to see more effort directed at establishing distributed trust methods for DRM, moving away from the centralized and tightly controlled frameworks that currently dominate the DRM landscape. Without such a shift, we will fail to see a solution that satisfies the competing but important needs of ubiquitous computing and DRM. **MM**

References

1. N. Davies and H.-W. Gellersen, "Beyond Prototypes: Challenges in Deploying Ubiquitous Systems," *IEEE Pervasive Computing*, vol. 1, no. 1, Jan. 2002, pp. 26-35.
2. G. Banavar and A. Bernstein, "Software Infrastructure and Design Challenges for Ubiquitous Computing Applications," *Comm. ACM*, vol. 45, no. 12, Dec. 2002, pp. 92-96.
3. J.P. Sousa and D. Garlan, "Aura: An Architectural Framework for User Mobility in Ubiquitous Computing Environments," *System Design, Development and Maintenance: Proc. Int'l Federation for Information Processing* (IFIP) *17th World Computer Congress and TC2 Stream 3rd Working IEEE/IFIP Conf. Software Architecture* (WICSA3), Kluwer Adademic Publishers, 2002, pp. 29-43.
4. S. Pearson, "Trusted Computing Platforms, the Next Security Solution," HP Labs, 2002.
5. E.W. Felten, "Understanding Trusted Computing: Will its Benefits Outweigh its Drawbacks?" *IEEE Security and Privacy*, vol. 1, no. 3, 2003, pp. 60-62.
6. A. Dornan, "Trusted Computing: A Matter of Trust," *Network Magazine*, vol. 19, no. 7, July 2004, pp. 26-32.
7. S. Ba and P.A. Pavlou, "Evidence of the Effect of Trust Building Technology in Electronic Markets: Price Premiums and Buyer Behavior," *MIS Quarterly*, vol. 26, no. 3, Sept. 2002, pp. 243-68.
8. W. Stallings, "PGP Web of Trust," *Byte*, vol. 20, no. 2, Feb. 1995, pp. 161-162.
9. G. Stix, "Some Rights Reserved," *Scientific Am.*, vol. 288, no. 3, Mar. 2003, p. 46.
10. K. Klemm and V.M. Eguíluz, "Growing Scale-Free Networks with Small-World Behavior," *Physical Rev. E* (*Statistical, Nonlinear, and Soft Matter Physics*), vol. 65, no. 5, May 2002, pp. 57102/1-57102/4.
11. P. Fergus et al., "DiSUS: Mobile Ad Hoc Network Unstructured Services," *Proc. Personal Wireless Comm.* (PWC 2003), Springer-Verlag, 2003, pp. 23-25.
12. M. Madden and L. Rainie, "Music and Video Downloading Moves beyond P2P," Pew Internet and American Life Project, 2005; http://www.pewinternet.org/pdfs/PIP_Filesharing_March05.pdf.
13. F. Zhou et al., "Approximate Object Location and Spam Filtering on Peer-to-Peer Systems," *Proc. ACM/IFIP/Usenix Int'l Middleware Conf.* (Middleware 2003), Springer-Verlag, 2003, pp. 1-20.



Madjid Merabti is director of the School of Computing and Mathematical Sciences, Liverpool John Moores University, UK. He also leads the Distributed Multimedia Systems and Security Research Group. His research interests are in distributed multimedia systems, including networks, operating systems, and computer security. Merabti has a PhD in computer science from Lancaster University.

David Llewellyn-Jones is a research associate at the Liverpool John Moores University. He is also part of the Distributed Multimedia Systems research group looking at secure component composition for personal ubiquitous computing. His research interests include computing and network security, component composition, and security frameworks in ubiquitous computing environments. Llewellyn-Jones received his PhD in model theory from the University of Birmingham.

Readers may contact Madjid Merabti at the James Parsons Building, Byrom St., Liverpool, L3 3AF, UK; M.Merabti@ljmu.ac.uk.