INTERTRUST®
STAR LAB

TECHNICAL REPORT STAR-TR-01-07

Supporting E-Commerce in Wireless Networks

RADEK VINGRALEK

October 2001

InterTrust Technologies Corporation
4750 Patrick Henry Drive
Santa Clara, CA 95054
Tel: +1.408.855.0100
Fax: +1.408.855.0136

Supporting E-Commerce in Wireless Networks

Radek Vingralek

STAR Lab, InterTrust Technologies Corporation
4750 Patrick Henry Drive
Santa Clara, CA 95054
rvingral@intertrust.com

Abstract. We describe a set of services that are either necessary or desirable for support of e-commerce transactions in wireless networks. We focus on e-commerce transactions that involve digitally represented information, such as audio or video files, computer software or text documents. For each proposed service we first explain its utility independently of the network connectivity and then we emphasize its benefits in wireless networks, when applicable.

1 Introduction

We describe a set of services in a wireless network infrastructure that are either necessary or desirable for support of e-commerce transactions. By e-commerce transactions we mean business transactions that are realized over a computer network. E-commerce transactions are specified by *contracts*, which define both the *obligations* and the *rights* of the buyer. For example, a contract may oblige the buyer to pay a fee before rendering a video file and give her a right to render the video file until an expiration date. We concentrate in this paper on e-commerce transactions that involve sale of digitally represented information, such as audio or video files, computer software or text documents. We call such e-commerce transactions *digital e-commerce transactions* (DETs). The enforcement of rights is particularly important for DETs, because digitally represented information can be easily copied and re-distributed. In the above example, the user may not be granted rights to create persistent copies of the video file. The systems that support DETs are typically called *Digital Rights Management* (DRM) systems.

The infrastructure services discussed in this paper do not depend on the type of communication network used to connect the buyers and the sellers. However, we will argue that the requirements are particularly important in wireless networks, which are distinguished by

- *Low bandwidth.* Most commercial wireless networks provide bandwidth in the range of 10 to 100 kbps, with most existing networks in the lower end of the spectrum and newer standards aiming at the higher end of the spectrum. The achieved bandwidth is further reduced by relatively high error rates.
- *High latency.* Although many wireless networks have similar latencies to fixed networks on the same scale, some have significantly higher latencies.

For example, the geosynchronous satellite networks have round trip latency of approximately 250 ms.
- *Intermittent connectivity.* Disconnections are more frequent in wireless networks than in fixed networks. For example, users of a cellular network may move to areas not covered by the network.

In the rest of this paper we describe the benefits of three services for support of DETs: 1) distributed trusted execution environment, 2) decentralized certificate infrastructure and 3) standardized contract languages.

2 Distributed Trusted Execution Environment

The Distributed Trusted Execution Environment (DTEE) ensures secrecy and integrity of a process state across computers in a network. In particular, the DTEE must protect the process state against a malicious disclosure or modification. DETs require process state integrity to guarantee that all obligations of a contract are satisfied and that the resulting rights are enforced on buyers' computers. DETs also require process state secrecy to prevent reverse-engineering and subsequent circumvention of the DTEE.

Consider again the video-on-demand system with a contract that obliges buyers to pay a fixed free before rendering the content and in exchange gives the buyer a right to play the content unlimited number of times prior to an expiry date. The integrity of the contract process state provided by the DTEE ensures that the buyer can get the video content only after paying the correct fee and that she cannot render the video after the expiry date. Secrecy of the contract process state provided by the DTEE protects sensitive data (such as keys used for encryption of the video) from disclosure to the buyer (the buyer may use the key to circumvent the DTEE and obtain the video for free).

DTEE's goals are complementary to the systems that enable secure execution of mobile code (such as Java applets). The Java virtual machine protects the integrity of the host system from a possibly hostile mobile code. The DTEE, on the other hand, protects the integrity mobile code state from a possibly hostile host system.

The DTEE must also ensure secrecy and integrity of operating system services used by the protected processes. A typical contract process may require access to persistent storage, time and location services. Trusted storage ensures that each contract process is given a private persistent storage that cannot be read by other processes. It may not be possible to prevent modification of the storage (ultimately, a buyer can destroy the storage device that is under her physical control), but the DTEE should detect such modification prior to reading from the storage. Contract processes may use persistent storage to store the values of counters. For example, a contract may allow a free rendition of a video for a limited number of times and require payment afterwards. Trusted time ensures that contract processes are given a reasonably accurate estimate of the current time or the time elapsed since a specific event. In some cases, the service may provide only a lower bound. For example, a contract may allow rendition of

a video until an expiry date. Trusted location gives to every process a reasonably accurate estimate of the current location of the hosting computer. For example, a video may be available at different prices depending on the country in which it is purchased.

2.1 DTEE Benefits

The DTEE is particularly useful for enabling DTEs in wireless networks, as demonstrated by the following examples:

Data prefetching. Wireless networks have a low bandwidth. Consequently, it is important to prefetch large data prior to their use. Some of the content files may be quite large, such as audio or video files. The content seller may distribute the content prior to its purchase (e.g., when the device is connected to a fast wired network or using a CD) and rely on the trusted storage to protect the content from consumption by the user until payment.

Message batching. Since communication is expensive in wireless networks, it is important to reduce also the overall number of messages. By using the trusted storage of the DTEE, contract processes may avoid communication with a remote computer during a DET. The payment in the above example can proceed without any message exchange with the seller's computer by keeping an audit trail of the transaction in the trusted storage of the buyer's device. The audit trail is periodically cleared with the payment infrastructure (e.g., the seller's computer) to limit the damage resulting from a loss of the buyer's computer and to free resources in the buyer's trusted storage. Periodic clearing also results in aggregation of several micropayments, which can reduce the per-transaction cost.

Fault tolerance. Since mobile computers may move to areas outside of the range of the wireless network, it is important that the computers can conduct (some) DETs even when disconnected from the network. In the above example, a buyer could purchase content without any message exchange. Therefore, the transaction could proceed even when the buyer's computer is disconnected.

The DTEE also enables new business models. Currently software is sold outright because it is difficult to control its usage once it has been transferred to the buyer's computer. The DTEE allows to replace the outright sale with a metered sale with a free preview for a limited number of times. Similarly, software can be leased for a limited period of time. Using a trusted location, software can be sold using different prices and business models in different countries or regions (which may be an effective way to combat software piracy).

Although most of the above examples utilize the DTEE to protect seller's interests on the buyer's computers, the DTEE can be also used in the reverse direction to protect buyer's privacy on the seller's computer. For example, a contract may provide a lower price in exchange for personal data. The buyer may bind the personal data with a (default) privacy contract, which would disallow redistribution of the personal data by the seller.

2.2 DTEE Implementation

Implementing the DTEE on general-purpose computers is difficult because it requires enforcing security policies on remote computers, which may be under control of an adversary. For example, a hostile user may load an emulator on her machine and gain a complete control over the execution of contract processes, which violates both the secrecy and integrity requirements on the DTEE.

However, unlike general-purpose computers, many wireless computers (such as high-end cell phones) are highly specialized devices that do not load user-provided programs. Consequently, the TDEE is vulnerable only to attacks on the hardware, which are more time-consuming than attacks on software [13] and the exploits are harder to distribute than software patches. The hardware can be further hardened against certain types of attacks using a single die systems in tamper-proof casing (e.g., smart-cards) [16], bus encryption [9] or tamper-resistant packaging coupled with memory zeroization on tamper detection [15].

On platforms that load user-provided programs (such as PDA's) the operating system must either load only programs from trusted sources and after their integrity has been validated [3, 15] or sandbox the untrusted processes [11, 12, 7]. Alternatively, the DTEE may settle for a weaker protection based on software obfuscation techniques [4, 8].

Trusted storage can be implemented on top of untrusted storage (e.g., a removable SmartMedia flash RAM card) using a short (e.g., 128 bit) secret memory, which can be read only by the contract process, and an one-way 32 bit counter, which cannot be decremented [14]. Trusted time can be implemented either by querying an authenticated NTP (Network Time Protocol) server or by using the trusted storage and a watch-dog timer, available on most embedded processors.

The protection provided by the DTEE does not have to be (and cannot be) perfect. It is sufficient to make the cost of a successful attack higher than the cost for the products purchased via the e-commerce infrastructure.

3 Decentralized Certificate Infrastructure

Since DETs involve interactions of multiple entities (*principals*) that may not know each other, it is important that the DRM system can use a *certificate infrastructure*, which provides means for *authentication* and *authorization* of all principals. Authentication establishes the validity of a public key representing a principal participating in a DET. Authorization establishes the capabilities of the principal owning the public key. Grants of capabilities to principals are expressed as *certificates*, which are signed with one or more private keys of the principals granting the capability.

In DETs, the capabilities involve access to digital content, clearing financial transactions, performing transaction escrow, and delegating capabilities to other entities (for example, a content provider may delegate transaction clearing capabilities to a partnering bank). Moreover, certificates may also attest an existence

(and quality) of a Trusted Execution Environment (TEE) on a given computer. The integrity of the DTEE can be maintained by revoking certificates of TEEs that are either known or suspected to be compromised. Similarly, a seller may require that the buyer's TEE has certain quality before it can participate in a contract execution.

Typical DRM systems involve a large number of principals, which can dynamically change. The principals may be authenticated by multiple authorities (e.g., a buyer may be authenticated by her ISP and a seller may be authenticated by one of commercial Certification Authorities). Moreover, principals may have a different degree of trust in authentication performed by other principals (e.g., principal A may trust authentication of principal B, but not C, while principal B may trust C, but not A). Therefore, the certificate infrastructure must be decentralized with a capability to delegate both authentication and authorization to other principals. In order to authorize principal A to perform action z, principal B must prove from the certificates it owns and the certificates supplied by principal A that the request conforms with principal B's local security policy. Several decentralized certificate infrastructures have been either specified or implemented as prototypes [6, 10, 5].

4 Standardized Contract Languages

There are several initiatives to standardize languages for expressing contracts [1, 2]. We see several benefits of adopting such a standard:

Multiple vendor DRM systems can interoperate. There are several aspects of interoperability: A buyer and seller can use different DRM systems. A seller can redistribute content from another seller (using a different DRM system) by augmenting the original contract. For example, an ISP can re-distribute videos as a value-added service. Similarly, a seller can embed content in a newly created content by including the contract of the embedded content in the newly created contract. For example, a news report may include photos from freelance reporters.

Buyers can find new content by querying on the terms of a contract. For example, a buyer may request a video that contains "Barney" in its title, costs no more than $2 and can be leased for at least one month. Moreover, the buyer may indicate that she is willing to provide restricted personal information (expressed using the contract language) in exchange for a free access. The DRM system evaluates the query and either returns the result set to the buyer for selection and/or transaction approval, or automatically purchases the best (or first) match.

Executing DETs as queries has additional benefits, particularly in wireless networks:

Asynchronous transaction execution. The entire transaction can be executed automatically with no involvement of the buyer (perhaps, except of the final approval). Consequently, the buyer does not have to manually search for a suitable product and execute individual steps of the contract, which

may lead to significant delays in wireless networks. For example, in the search step the user may need to download descriptions of different products that may be represented as large graphical objects.

Transaction shipping. The query can be shipped for evaluation to the sellers' computers, which can further reduce the volume of data that is sent across the wireless network (in fact, the decision whether to ship the query to the data or the data to the query can be cost-based, but we assume that for most DETs the former is better).

References

1. The open digital rights language initiative. `odrl.net`, September 2001.
2. Xrml. `www.xrml.org`, September 2001.
3. W. Arbaugh, D. Farber, and J. Smith. A secure and reliable bootstrap architecture. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1997. Oakland, CA.
4. D. Aucsmith. Tamper resistant software: an implementation. In *Proc. International Workshop on Information Hiding*, 1996. Cambridge, UK.
5. M. Blaze, J. Feigenbaum, J. Ionnidis, and A. Keromytis. The KeyNote trust management system, version 2 (rfc 2704), 1999.
6. M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1996. Oakland, CA.
7. F. Chang, A. Itzkovitz, and V. Karamcheti. User-level resource-constrained sandboxing. In *Proceedings of the 4th USENIX Windows Systems Symposium*, 2000. Seattle, WA.
8. C. Collberg, C. Thomborson, and D. Low. Manufacturing cheap, resilient, and stealthy opaque constructs. In *Proceedings of the ACM Conference on Principles of Programming Languages*, 1998. San Diego, CA.
9. Dallas Semiconductor. *DS5002FP Secure Microprocessor Chip*, July 2001.
10. C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI certificate theory (rfc 2693). `ftp://ftp.isi.edu/in-notes/rfc2693.txt`, 1999.
11. I. Goldberg, D. Wagner, R. Thomas, and E. Brewer. A secure environment for untrusted helper applications. In *Proceedings of the 6th USENIX Security Symposium*, 1996. San Jose, CA.
12. G. Hunt and D. Brubacher. Detours: Binary interception of Win32 functions. In *Proceedings of the 3rd USENIX Windows NT Symposium*, 1999. Seattle, WA.
13. O. Kommerling and M. Kuhn. Design principles for tamper-resistant smartcard processors. In *Proceedings of the USENIX Workshop on Smartcard Technology*, 1999.
14. U. Maheshwari, R. Vingralek, and W. Shapiro. How to build a trusted database system on untrusted storage. In *Proceedings of the 4th Symposium on Operating Systems Design and Implementation*, 2000. San Diego, CA.
15. S. Smith, E. Palmer, and S. Weingart. Using a high-performance, programmable secure coprocessor. In *Proceedings of the International Conference on Financial Cryptography*, 1998. Anguilla, British West Indies.
16. J. Tual. MASSC: A generic architecture for multiapplication smart cards. *IEEE Micro*, 19, 1999.