

Distribution Chain Security

Glenn Durfee^{*}
Computer Science Department
Stanford University
Stanford, CA 94305-9025
gdurf@cs.stanford.edu

Matt Franklin^{*}
Department of Computer Science
University of California, Davis
Davis, CA 95616-8562
franklin@cs.ucdavis.edu

ABSTRACT

Digital content distribution systems will enable business models in the near future that cannot be predicted today. In this paper, we identify a new security problem that can be crucial to this enablement. The problem arises from the conflicting privacy and integrity goals of middlemen in digital distribution chains. Our solution is a novel system design that incorporates obfuscated digital contracts, semi-trusted contract certifiers, and zero-knowledge proofs of arithmetic relations. Our implementation and timing experiments demonstrate that our solution is practical and efficient.

Categories and Subject Descriptors

K.4.4 [Computers and Society]: Electronic Commerce—*intellectual property, payment schemes, security*; K.5.1 [Legal Aspects of Computing]: Hardware/Software Protection—*proprietary rights*

General Terms

Security

Keywords

Digital distribution chains, digital property rights

1. INTRODUCTION

In recent years, a number of digital contract systems have been proposed to encourage the production, distribution, and sale of high-quality digital works on the Internet. These systems allow rights, fees, and other terms and conditions to be associated with a given work. Combined with proposed secure software and hardware, digital contracts can be enforced to insure proper payment and rights management for digital works. Of course, there are many potential vulnerabilities at the software and hardware levels that an

^{*}Work done while at Xerox Palo Alto Research Center.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CCS '00, Athens, Greece.

Copyright 2000 ACM 1-58113-203-4/00/0011 ..\$5.00

attacker could exploit to subvert a digital contract system. We will not address those kinds of security issues in this paper. Instead, we will examine a new kind of security problem that can arise even when the system is assumed to protect content and enforce contracts as specified.

Systems to enforce digital contracts are already in place or will be available soon – see [14, 15, 16, 13] for a few examples. Initially, these kinds of systems might be used to support relatively simple business models. Specifically, we would not expect to see very early support for business models that involved *middlemen*. Middlemen are entities that are neither direct producers nor direct consumers of digital content. Instead, they might buy digital content from one or more providers, repackage or otherwise add value, and then resell to consumers or to other middlemen.

Every middleman in a distribution chain produces a new contract to enforce a new set of terms and conditions. For long distribution chains, it may be unwieldy to carry along every contract that was created along the way. This would be undesirable from a practical business perspective as well. The seller has a strong incentive to conceal from the buyer earlier contracts along a chain. If a potential buyer gets to see earlier contracts, the buyer might be tempted to renegotiate the new contract at more favorable terms, or “disintermediate” the seller and negotiate independently with earlier sellers in the chain. There is a natural privacy requirement here that may be crucial for creating viable business niches.

There is a natural integrity requirement that is also fundamental. A middleman may wish to keep secret the favorable terms of earlier contracts in a distribution chain, but he must not be allowed to negotiate a new contract that violates his existing obligations. As a result of adding value to a digital work, a reseller is certainly entitled to be compensated by raising the fees associated with use of the content. Other terms associated with the earlier contract might be altered by a reseller as well. Expiration dates might be moved up, or hardware security requirements might be tightened, or payment options might shift (e.g., from flat rates to per-use fees). Any changes that are made in the new contract must in some sense be “faithful” to the original contract – existing payment obligations must not decrease, expiration dates must not move into the future, new rights cannot be granted, and so on.

These two requirements are in conflict with each other. Verifying that a new contract is faithful to an old contract seems to require that both contracts be inspected by the new buyer, but that would violate the privacy needs of the seller.

In this paper, we will explore a novel approach to balancing distribution chain privacy and distribution chain integrity. Intuitively, our solution is to present versions of each contract to a contract certifier in which sensitive fields have been “obfuscated” (encrypted, or committed to). In order to verify the “faithfulness relation” between contracts, a “proof of faithfulness” must accompany the obfuscated versions of the contracts. But such a proof must not reveal anything meaningful about the obfuscated values. Since efficiency is crucial, we implemented our solution to test its effectiveness. In this way, we demonstrate an extremely efficient solution to the problem of distribution chain security, and thus extend the realm of real-world settings for which zero-knowledge proof techniques are practical.

Related Work: There has been a lot of work to develop protocols for the fair exchange of digital content and digital signatures. See [4, 3] for important early work on these problems, and [1, 2] for more recent developments. Despite the common focus on “contracts”, our work is quite distinct. The work on fair exchange is typically concerned with atomicity, i.e., that an exchange benefits both parties or neither, and secrecy is considered only as an intermediate means to that end. It is possible to incorporate fairness mechanisms to our protocols, but we do not address that interesting question in this paper.

Organization of Paper: The rest of the paper is organized as follows. In Section 2, we discuss the requirement of distribution chain integrity. The requirement of distribution chain privacy, and an overview of our cryptographic approach, is treated in Section 3. Section 4 discusses special issues for distribution chain security when financial transactions are involved. The system design is discussed in Section 5. Implementation details and performance measurements are given in Section 6. Alternative models are considered in Section 7. Conclusions and open problems are mentioned in Section 8.

2. DISTRIBUTION CHAIN INTEGRITY

If resale and distribution of digital works is to be allowed, safeguards must be provided to ensure that existing obligations to rights holders of resold works are maintained. In this section we provide an outline of relevant terms of this “faithfulness” requirement for an example property rights language. The approach taken here applies to large class of rights languages.

The following terms embody concepts common to all property rights languages, and are used throughout this document.

Right: A right specified by a contract is a term or sequence of keywords in a contract, the presence of which in the contract allows a holder of the work to use it in a specified manner. For example, the right to print or to sell a copy of the work may be granted to a user. Furthermore, a right may have conditions, fees, and terms associated with it which must be satisfied in order to exercise in the specified activity. For example, the right to print the content may be associated with the requirement to pay a certain sum of money, along with the requirement that the work may be printed only using a trusted printer capable of embedding watermarks.

Examples of rights include printing, rendering, playing, copying, selling, and loaning the work, as well as including

excerpts of the content in other works and embedding the content into other works. Specifications and terms may include fee specifications, time specifications (such as release and expiration dates), hardware security requirements, and watermarking requirements.

Contract: A contract is a specification written in a property rights language, such as the Digital Property Rights Language [14] of the rights granted to the holder of the digital work and the terms associated with exercising those permissions. Literally, it is a sequence of string tokens conforming to the rights language syntax, stored as a file and linked to the corresponding work in a way that forbids modification by users. (A number of methods have been proposed to accomplish this, including use of secure hardware or software, cryptographic techniques, or secure contract servers. The particular method used is unimportant for our purposes.)

Author: The author of a digital work is the original creator and can specify (possibly in conjunction with a publisher) the initial contract to be associated with a work.

Reseller: A reseller, such as a publisher, distributor or other middleman is an entity intermediate in the chain of distribution. A reseller makes a work available after modifying it, bundling it with other content, or performing some other service with respect to the work or its distribution. In doing so the reseller typically replaces the contract associated with the work.

2.1 Ensuring Integrity

When a reseller wishes to change the rights or obligations associated with a work, it must be confirmed that these changes do not violate the terms of the original contract. For example, legal obligations to pay existing royalties cannot be erased, and these amounts cannot be decreased, although perhaps the form of payment can be altered. When contract C_{new} honors the terms of contract C_{old} , we say that C_{new} is faithful to C_{old} , or that the predicate of faithfulness holds.

Each rights language L gives rise to a particular set of contracts and a particular faithfulness relation. We conjecture that for most contract languages, the predicate of faithfulness between contracts can be expressed using arithmetic expressions between the terms of C_{old} and C_{new} . An important open question is whether this is true for every contract language. Here we outline an example of how this relation looks for a large generic class of rights languages.

A general condition that must hold is that if the new contract C_{new} grants a right R to a buyer or user of the work, then that right R must be granted in the original contract C_{old} . Furthermore, except for the terms listed below, every term associated with R in C_{new} must match those in C_{old} exactly. The exceptions are as follows:

Expiration and release dates: If contract C_{old} specifies that a right R expires on a date D , then the contract C_{new} must specify that the right R expires on a date D' , where $D' \leq D$ (D' is at least as soon as D). Similarly, if D and D' encode release dates (earliest possible date in which the right can be invoked), then we require $D' \geq D$.

Fee specifications: In the simplest case, if contract C_{old} specifies that exercising a right R includes an obligation to pay party P an amount A , then the contract C_{new} must include an obligation to pay party P an amount A' , where $A' \geq A$, for exercising the same right R . More complicated but easily analyzable examples exist in which payment

amounts are converted to and from metered rates, per-use fees, and flat fees, and between currencies.

This assumes a the financial model in which a user pays the single fee A' to the single entity P (such as a financial clearinghouse) when a right is executed. Then P is responsible for dividing A' among deserving parties, resulting in the simple condition that $A \leq A'$. In this example, however, it is clear that the financial clearing house can now learn a great deal about the financial lifecycle of a particular digital work, which in many cases might represent an undesirable loss of privacy. However, there are many other examples of payment models that can be used that do not suffer from this weakness. We postpone discussion of these other schemes to section 4 until we have had a more complete review of privacy.

Security access requirements: If contract C_{old} specifies that exercising a right R requires trusted hardware of a security level L , then contract C_{new} must specify that exercising right R requires trusted hardware of security level L' satisfying $L' \geq L$.

3. DISTRIBUTION CHAIN PRIVACY

In the last section we outlined examples of details that must be checked to ensure integrity when associating new contracts with digital works. In this section we outline a protocol that allows the details about old and new contracts to be kept secret, while still enabling a proof of the integrity of the system to be provided.

The system we propose is based on a system for creating and verifying zero-knowledge proofs [10] of predicates in a commitment scheme. An important open question is whether or not the predicates of interest, that is, predicates expressing a faithfulness relation between contracts can be efficiently represented using existing zero-knowledge techniques. In our solution, this is the case.

To describe our solution we must first define the following terms.

Commitment: A commitment can be thought of as a “sealed envelope” which cannot be opened without knowing the value inside the envelope. If $c = \text{commit}(v)$ then it should be difficult or impossible to compute v directly from c . We refer to this as the *hiding* property of the commitment scheme.

If a commitment c and a candidate value v is supplied, it should be easy to check if $c = \text{commit}(v)$. We refer to this as the *verifiability* property of a commitment scheme.

It should be difficult or impossible to create a commitment for which two different values could be successfully revealed; that is, given any c and v satisfying $c = \text{commit}(v)$, it should be difficult to produce a $v' \neq v$ such that the equality $c = \text{commit}(v')$ holds. We refer to this as the *binding* property of the scheme.

The commitment schemes described in [7] satisfy all three of these conditions under suitable intractability assumptions.

Zero-Knowledge Proof: A zero-knowledge proof Z of a predicate P on values v_1, \dots, v_n is a message from a *prover* (usually a reseller) to a *verifier* (which for our purposes will be the contract certification authority, although it could also be a user, publisher, author, or another reseller) intended to convey the validity of the predicate P without revealing the values v_i . We restrict attention to zero-knowledge proofs

that have been made non-interactive using the Fiat-Shamir heuristic [9].¹

In practice, a verifier of the proof receives $\text{commit}(v_1), \dots, \text{commit}(v_n)$ along with the proof Z . The verifier performs a computation to determine if the proof is valid, and should accept the proof (thus believing that P holds) with probability exponentially close to 1 if P in fact holds. The verifier should learn no information about the values v_i except that they satisfy the predicate P .

On the other hand, if the predicate P does *not* hold on the values v_i then there should be at most an exponentially small probability that the prover can create a “proof” that fools a verifier. Furthermore, a proof for particular values v_i cannot be manipulated in a way to allow it to verify for any other values $v'_i \neq v_i$.

Obfuscated Contract: An obfuscated contract is a contract in which some of the values have been replaced by commitments to those values. If C is a contract with values v_1, \dots, v_n that we wish to obfuscate, we may produce obfuscated contract $\text{obs}(C)$ by replacing each value v_i with the commitment $\text{commit}(v_i)$. The obfuscation of contracts can be thought of as a commitment scheme for contracts.

In practice, only the handful of values corresponding to sensitive numeric or string fields will be replaced with commitments. Therefore, an obfuscated contract differs only very slightly in length and structure from its unobfuscated counterpart.

Contract Certifier: The contract certifier is a corporate structure or a legal entity in charge of certifying newly-created contracts. We also use the term “contract authorizer”. Typically, this certification is achieved using a digital signature; that is the example that we will assume is in use, although these techniques work with any number of methods. If a contract C or obfuscated contract $\text{obs}(C)$ has been digitally signed by the contract certifier, it will be denoted by $[C]$ or $[\text{obs}(C)]$, respectively. We will refer to this contract or obfuscated contract as *approved*, *certified*, or *signed*.

3.1 Ensuring Privacy

Here we fix a particular scenario and demonstrate a scheme that can be employed to ensure both distribution chain integrity and privacy. We assume that buyers (or their software) will accept only contracts which have been approved by the contract certifier. To maintain privacy, the contract certifier will see only obfuscated contracts, so only obfuscated contracts can be directly signed by the authority. Therefore, approval of a contract C is ascertained by providing the buyer or user with an obfuscated contract $[\text{obs}(C)]$ that has been digitally signed by the certification authority, along with the same contract C with terms “in the clear”. The software checks that the contract C matches the obfuscated contract $\text{obs}(C)$ (using the verification property of commitment schemes described above) and checks the digital signature on the obfuscated contract $[\text{obs}(C)]$. If these

¹This heuristic “flattens” any interactive protocol where the verifier’s challenges are random coin flips. The prover computes all of the challenges himself deterministically by applying a cryptographically strong hash function to earlier messages and inputs of the protocol. Under the heuristic assumption that the hash function behaves like a random oracle, these challenges are generated in a manner that is unpredictable and uncontrollable, and so the prover cannot gain an advantage.

tests succeed, then based on the binding property of the commitment scheme, the user knows that the presented contract C uniquely matches the certified version $[\text{obs}(C)]$. See Figure 1.

If presented with a digital work by its author along with an obfuscated version $\text{obs}(C)$ of the proposed contract C , the contract certifier checks that the obfuscated contract is well-formed (i.e., is syntactically valid in the rights language) and that the values supplied for each entry is within any system-wide bounds for the fields of the appropriate type that might be in place. Checking the latter condition may require zero-knowledge proofs for the boundedness of certain fields. If these conditions hold, the contract certifier digitally signs the obfuscated contract and returns $[\text{obs}(C)]$ to the author.

A reseller, like any other buyer in this system, obtains a digital work with a digitally signed, obfuscated contract $[\text{obs}(C)]$ and the corresponding (unsigned) contract C in the clear. To obtain certification for a new contract D on a digital work, it must convince the contract certifier of the faithfulness of the new contract. To do this, the reseller creates $\text{obs}(D)$ and a zero-knowledge proof of faithfulness Z establishing that D is faithful to C . It sends the contracts $[\text{obs}(C)]$ and $\text{obs}(D)$ and the zero-knowledge proof of faithfulness Z to the contract certifier. If indeed D is faithful to C , then the proof Z will verify correctly and the contract certifier will return the digitally signed $[\text{obs}(D)]$ to the reseller to give to buyers or other entities in the chain of distribution. See Figure 2.

Note that once the proof of faithfulness has been created and used to convince the contract certifier of the faithfulness of the new contract, it is no longer needed. Only the new contract D and the new certified obfuscated contract $[\text{obs}(D)]$ are needed for the next step in the chain of distribution. This is an important optimization step in our scheme.

3.2 Creating a Proof of Faithfulness

A proof of faithfulness is constructed as follows. The obfuscated contracts $\text{obs}(C)$ and $\text{obs}(D)$ consist of sequences of tokens in the rights language, some of which represent commitments to secret values. Because the reseller has access to C and D , it knows how to open each of these commitments. The aim is to prove that the values contained in matching commitments are in the proper relationships.

For example, the reseller might wish to provide a proof of the predicate $P(\alpha, \beta) \equiv \alpha \leq \beta$ to show that the commitment in D containing the fee β opens to a value no

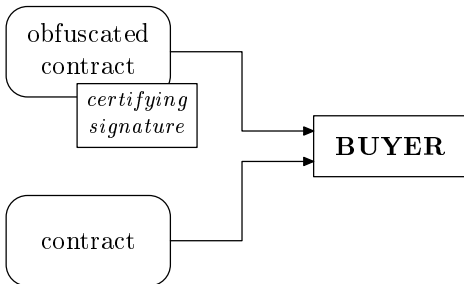


Figure 1: Buyer receives contract and certified obfuscated contract and verifies these match.

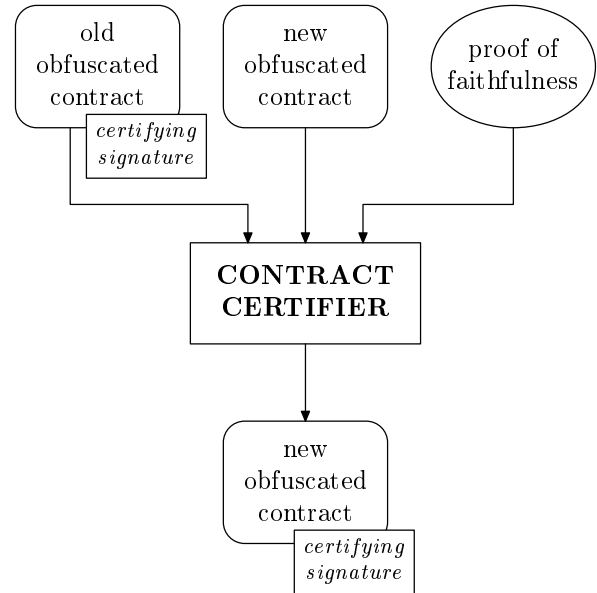


Figure 2: Contract certifier checks proof of faithfulness for contracts before issuing new certification.

less than the corresponding commitment in C containing α . In general, all predicates that must be shown are compositions of elementary predicates for equality ($\alpha = \beta$), addition ($\alpha + \beta = \gamma$), multiplication ($\alpha\beta = \gamma$), inequality ($\alpha \leq \beta$), and bit-value ($\alpha \in \{0, 1\}$), or compositions and boolean combinations of these predicates.

Fortunately, it is easy to *efficiently* generate short proofs of these predicates using the commitment scheme of Cramer-Damgård. Their paper [7] describes how to implement equality, addition, multiplication, and bit-value proofs of constant size. Inequality requires slightly more work. This is discussed in more detail in Section 5.

4. DISTRIBUTION CHAIN PAYMENT

One important area of study is how different choices of fee payment schemes affect the resulting faithfulness predicates and zero-knowledge proofs. For instance, instead of listing a single fee and account number in the digital contract, one can maintain a list of fee-payee pairs. Then instead of routing funds from a user to a clearinghouse in a single transaction, the user is charged the sum of the fees and the software routes the shares of that sum individually to the clearinghouse on behalf of each payee (or the shares are sent to the payees directly.)

While this scheme allows the privacy of the reseller's profit margin to be maintained with respect to the contract certifier, this information is now unfortunately revealed to the buyer, or at least the buyer's software. In most cases this is unacceptable, but the following enhanced version of this scheme can be used to keep this information private from the end user as well. Suppose the banking system employs an encryption scheme that supports blinding.² Each contract contains a list of (fee, encrypted account number) pairs. The

²A cryptosystem supports blinding when it is possible for any user, given an encryption $y = E_{p_k}(x)$, to find a $y' \neq$

bank receives funds from the user along with the encrypted account numbers; these are decrypted and funds routed to the appropriate payees. Additional privacy can be gained by using the blinding property of the encryption scheme: when fees are added to a new contract, every existing payment obligation is “split” into fees the size of their least common multiple, or some minimum value; the account numbers corresponding to these fractional fees are blinded, to keep the total profit of any one recipient secret from the contract certifier and buyers; and the entire list is permuted, to keep the order in which fee increases were made secret.

In order to prove that fee obligations are maintained, the reseller provides the contract certifier with the permutation used in the last step, the blinding factors used to anonymize the account numbers, and a proof that the sums of the fractional fees are not less than the corresponding fees in the original contract for each original account. (Note that while the fees are obfuscated, the account numbers are already encrypted and therefore it is not necessary to further obfuscate them.) This proof can be provided efficiently as it can be represented using simple arithmetic relations.

Another scenario is one in which the total fee is listed in the digital contract and the last seller in the chain of distribution is listed as the recipient of those funds. In addition, however, a list of electronic addresses of other payees is maintained, with no information about the size of their shares given. Payment is routed to the last reseller and a notice is given to the remaining parties; these can either generate requests from these parties to the reseller to obtain their shares of the funds, or be logged for accounting purposes or later retrieval. In this scenario, although the number and identities of middlemen in the chain of distribution is revealed, the privacy of the profit margin is maintained from not only the contract certifier and the end user, but the financial clearinghouse as well.

It may be desirable to allow resellers to lose money on individual transactions, e.g., by repackaging attractive offers as “loss-leaders”. Some fee payment schemes can support this in our approach, when combined with appropriate efficient zero-knowledge proofs of range-boundedness, sums, and less-than predicates.

5. SYSTEM DESIGN

While it is well-known that generic zero-knowledge techniques can be used to prove any predicate which can be evaluated in polynomial time “in the clear” [11], those results are mainly theoretical and have been inefficient when implemented in practice for specific problems. What we show in this section is that generic zero-knowledge techniques can indeed be quite efficient and practical when applied to the problem of distribution chain security. First we describe the choices of commitment schemes and ramifications on the system. Then we outline our technique for proving the most complicated of predicates needed, the less than or equal to predicate. Finally, we describe the performance of our implementation of these zero-knowledge proof

y that is also a valid encryption of x , i.e., $x = D_{s_k}(y')$. This should be possible even when the user does not know the secret key s_k . Furthermore, given y, y' , it should be infeasible to those without s_k to determine whether or not they are encryptions of the same message. The well-known ElGamal public key encryption scheme [8] is an example of a cryptosystem that supports blinding.

generation and verification techniques when used to obtain distribution chain security.

5.1 Choice of Commitment Scheme

Four commitment schemes are summarized in [7]; the following two are of interest to us. The first is a scheme based on the difficulty of the discrete logarithm problem in subgroups. In this scheme, the contract certifier chooses primes p, q such that q is a divisor of $p - 1$, with p much larger than q . (It is typical to have p be 1024 bits in length and q be 160 bits in length.) Values $g, h \in Z_p$ are chosen such that g and $h = g^x$ generate the same subgroup of Z_p order q . The contract certifier keeps the value x secret and announces the values p, q, g and h publicly. These will be needed by the author and resellers to generate obfuscated contracts.

To generate a commitment to the value α , one computes $c = g^\alpha h^r \pmod p$ for a randomly-chosen value r . Using the above parameters this commitment will be 1024 bits in length. Commitments in this scheme enjoy *unconditional hiding* – even with unlimited computational power it is impossible for any party to determine the value α from c . Therefore, distribution chain privacy is complete. However, these commitments are *conditionally binding* – we are guaranteed that a party can open the commitment c to reveal only the value α only if the party has not obtained the value x . The buyer, then, must trust that the contract certifier’s secret x has not been compromised by the seller to ensure distribution chain integrity.

The second commitment scheme is based on the difficulty of the Decision Diffie-Hellman problem in subgroups. In this scenario, the author generates the parameters p, q, g, h as above. A commitment to a value $\alpha \in Z_q$ is expressed as the pair

$$c = (g^{\alpha+r}, h^r)$$

for a randomly-chosen $r \in Z_q$. Commitments in this scheme, using the above parameters, will be 2048 bits in length. Unlike the first scheme, these commitments enjoy *unconditional binding*, which guarantees that a commitment can be opened to one value only. This ensures complete distribution chain integrity. However, this scheme are *conditionally hiding* – if the contract certifier obtains the author’s secret x (which may be revealed to resellers as well, although there is no advantage in doing so) then distribution chain privacy cannot be ensured.

Because the two schemes differ in the hiding and binding properties, the proper choice of commitment scheme depends on the security model being used. However, in most cases the first scheme is sufficient, and is preferred due to its use of smaller commitments and more efficient proof generation and verification steps. To prevent against compromise of the contract verifier’s secret x , this value can be erased immediately after $h = g^x$ is computed, as x is not needed for any of our protocols.

Remark. It should be noted that these commitment schemes use as additional input a random value r . To include this in our scheme, we suggest for compactness that the random value r be determined by the output of a pseudorandom number generator or pseudorandom function. A seed R to this generator should be included in the unobfuscated contract; to generate the i th commitment c_i , the reseller can compute the i th output r_i from the pseudorandom generator seeded with r and compute $c_i = \text{commit}(v_i, r_i)$. A buyer

can confirm that the i th commitment matches a value in the clear by extracting R from the unobfuscated contract, computing r_i , and checking $c_i = \text{commit}(v_i, r_i)$.

5.2 Choice of Zero-Knowledge Proofs

For our implementation, we use the efficient zero-knowledge proof schemes of Cramer-Damgård [7]. For both of the commitments schemes described above, they give efficient proofs for equality, arbitrary sums, pairwise products, and equality to zero or one. Actual proof sizes and running times for generation and verification is discussed in Section 6. Intuitively, much of the efficiency derives from the “homomorphic” properties of the underlying commitment schemes. For example, the product of two commitments is a commitment of the sum of the committed values. These schemes also use the standard Fiat-Shamir heuristic [9] to make the proofs non-interactive. The details of these proofs are omitted (see [7]).

To create proofs for predicates involving \leq , we develop a new technique to improve on the efficiency of published schemes. Our solution is similar to an independently and previously discovered unpublished technique by Schoenmakers. (See also [12].) For simplicity, we assume we are using the discrete-log-based commitment scheme described above, although the approach works for the other commitment scheme as well.

Suppose we wish to prove $\alpha \leq \beta$ for values $\alpha, \beta \in [0, 2^t)$ and $2^t < q/2$. If $\gamma = \beta - \alpha$, then we know that $\gamma \in [0, 2^t)$ if and only if $\alpha \leq \beta$. Thus our aim is to create an efficient zero-knowledge proof of this bound on γ . In fact, our proof will be the size of about $\log \beta$ commitments.

We are working with commitments $a = \text{commit}(\alpha, r) = g^\alpha h^r$ and $b = \text{commit}(\beta, s) = g^\beta h^s$ for some random $r, s \in \mathbb{Z}_q$. The prover computes the commitment

$$c = \text{commit}(\gamma, r - s),$$

which the verifier can independently compute as $c = a/b$.

Let $(\gamma_i)_{i=0}^{t-1}$ be the binary representation of $\beta - \alpha$, i.e., $\beta - \alpha = \sum_{i=0}^{t-1} 2^i \gamma_i$. Choose random values r_1, \dots, r_{t-1} and set $r_0 = r - s - \sum_{i=1}^{t-1} 2^i r_i$. Then let $c_i = \text{commit}(\gamma_i, r_i) = g^{\gamma_i} h^{r_i}$ for all $i = 0, \dots, t-1$.

If the verifier knows that the bound $\alpha, \beta \in [0, 2^t)$ holds, then it is sufficient to provide the commitments c_0, \dots, c_{t-1} along with a proof that each c_i is a bit. The verifier checks each c_i bit proof and confirms that indeed $b \prod c_i = a$.

If the verifier does not know that α and β are bounded, then a zero-knowledge proof that $\alpha \in [0, 2^t)$ and $\beta \in [0, 2^t)$ can be constructed in a similar way and must accompany the above proofs. However, it is usually the case that one of the values comes from an obfuscated contract that has already been certified, and therefore the certifier trusts that it is bounded. In that case, we may optimize the proof size by leaving out the unnecessary proof.

We note that a technique developed recently by Boudot [6] can be used to generate *constant*-size proofs that α, β , and $(\beta - \alpha)$ are bounded, at the expense of using a larger modulus for commitments. This technique becomes more efficient to use than the above if the bounds on the relevant values are very large; for our purposes, however, the above technique is usually more efficient, and is much easier to implement.

More complicated arithmetic predicates can be expressed through the use of commitments to temporary values. For example, to provide a proof of $P \equiv \alpha\beta + \gamma \leq \delta$, we may cre-

ate temporary commitments to values ϵ, ζ and show $\alpha\beta = \epsilon$, $\epsilon + \gamma = \zeta$, and $\zeta \leq \delta$. The proof of the predicate P then consists of the temporary commitments followed by proofs for each of the elementary arithmetic predicates. This composition is clearly zero-knowledge. Furthermore, predicates that are boolean combinations of these predicates can be represented with little extra overhead using standard techniques.

6. IMPLEMENTATION AND TIMING DETAILS

Using the discrete-log-based commitment scheme outlined above, the proof method outlined above produces fairly small proofs. For example, suppose contracts C and D each contain a sensitive field such as a time or a price. A typical size for a commitment is 1024 bits, or 128 bytes. This yields obfuscated contracts $\text{obs}(C)$ and $\text{obs}(D)$ with corresponding fields of size 128 bytes.

To prove that obfuscated commitments are in the correct relationship to yield a faithful contract, we might require one or some combination of the following primitive proofs. We might need a proof that the values that two commitments encode are equal; using the parameters expressed above, this will require 148 bytes. A proof that the value encoded in a commitment is a bit uses 316 bytes. A proof that the value in a commitment is the product of two the values encoded in two other commitments will require 484 bytes. To generate a proof that a committed t -bit value is less than another committed t -bit value requires t additional commitments and t bit proofs; for the typical value of $t = 20$ (the size of a field storing a date or a price), this would yield a proof size of about 9 kilobytes. An additional 9 kilobytes is needed for each operand to prove that it is a t -bit value; however, usually only one such extra proof is required per inequality.

Suppose a contract has 15 sensitive fields. The obfuscated version of the contract will typically be 2-3K larger than the unobfuscated version. If five of these fields require equality proofs, five require multiplication proofs, and five represent 20-bit quantities requiring less-than proofs, the total “proof of faithfulness” will be approximately 85K. (About 80K of this is needed to store the less-than proofs, which clearly dominate the space requirements in this system.)

We ran our implementation in Java on a 450 MHz Intel Pentium III running Linux and the Blackdown implementation of the Java 1.2 JDK. With p of size 1024 bits and q of size 160 in the discrete-log-based scheme, creating commitments of arbitrary 160-bit values takes approximately 50 milliseconds. Creating commitments to small values, which is typical in these contracts, takes approximately 25 milliseconds. Creating equality proofs takes approximately 25 milliseconds, while verifying equality proofs takes about 50 milliseconds. Additions do not need proofs since in the scheme a commitment to the sum of k committed values can be computed as the product of the k commitments. (So to prove $\alpha + \beta = \gamma$, only one equality proof is needed.) Proving that α is either 0 or 1 requires 80 milliseconds, while verifying this proof takes about 100 milliseconds. Creating multiplication proofs requires about 130 milliseconds, while verifying multiplication proofs requires approximately 200 milliseconds. Creating and verifying “less than or equal to” proofs for 20-bit quantities each require about 1.8 seconds. These running

proof type	size	create time	verify time
commitment	128 bytes	25-50 ms	25-50 ms
$\alpha = \beta$	148 bytes	25 ms	50 ms
$\alpha + \beta = \gamma$	148 bytes	25 ms	50 ms
$\alpha \in \{0, 1\}$	316 bytes	80 ms	100 ms
$\alpha\beta = \gamma$	484 bytes	130 ms	200 ms
$\alpha \leq \beta$	9 kilobytes	1.8 s	1.8 s

Figure 3: Running times and proof sizes on a 450 MHz Intel Pentium III running Linux. Security parameters chosen were $p = 1024$ bits and $q = 160$ bits.

times are dominated by the time needed to perform modular exponentiation of a 1024-bit base to a 160-bit exponent, which, using the Java large integer package, takes approximately 20-22 milliseconds. These times are summarized in Figure 3.

7. OTHER MODELS

The use of a contract certifier is just one example of a scenario which can take advantage of zero-knowledge proofs of faithfulness. Other scenarios are possible; in a scenario with no contract certifier, a reseller would append changes to work's original obfuscated contract along with a proof of faithfulness of these changes. The concatenation of obfuscated contracts, corresponding proofs of faithfulness, and (only) the final plaintext contract would be revealed to the buyer.

Although this scheme has the advantage that it does not need a third party to certify contracts, it suffers from very large contract sizes, since the contract grows substantially each time changes are made. The advantage of contract certification is that it allows us to “throw away” the proof of faithfulness; that optimization is not possible in this scenario. Furthermore, contract verification is time consuming, as the entire composition of changes needs to be verified. It is precisely these compositions that are optimized by the contract authorizer. Lastly, for this scheme to work it is necessary to use a commitment scheme with unconditional binding, like the one outlined in Section 5.1.

8. CONCLUSION AND OPEN QUESTIONS

In conclusion, the problem of distribution chain security is a new and important issue in digital content distribution systems. We have identified this new problem, and proposed a cryptographic solution based on commitments and zero-knowledge proofs of arithmetic relations. Our implementation and timing experiments indicate that this solution is practical and efficient.

A number of other questions remain that deserve further study. For instance, a rich class of very expressive contract languages is currently being developed, and while the predicates discussed here capture a large number of faithfulness relations, there remain a number of exceptional cases to be studied. Each contract language gives rise to a notion of faithfulness between contracts. In general, the set of contracts along with the faithfulness relation forms a lattice. A formal model of contract faithfulness along these lines would be helpful. It would also be interesting to study exactly which relations in this “lattice of faithful contracts” can be expressed efficiently using zero-knowledge predicates.

In the simple scenario presented here, it is easy to prove the zero-knowledge properties of the protocols used. Because we would like to use these techniques to encompass a wide variety of scenarios, it would be beneficial to have a formal model to express the knowledge, trust, and privacy characteristics of the entities involved.

9. REFERENCES

- [1] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In proceedings *ACM Conference on Computer and Communication Security*, pp. 6–17, 1997.
- [2] G. Ateniese. Efficient verifiable encryption (and fair exchange) of digital signatures. In proceedings *ACM Conference on Computer and Communication Security*, pp. 138–146, 1999.
- [3] M. Blum. Three applications of the oblivious transfer. Part I: Coin flipping by telephone; Part II: How to exchange secrets; Part III: How to send certified electronic mail. Department of EECS, University of California, Berkeley, 1981.
- [4] M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, vol. 36, pp. 40–46, 1990.
- [5] E. Brickell, D. Chaum, I. Damgård, and J. van de Graaf. Gradual and verifiable release of a secret. In proceedings *Crypto '87*, Lecture Notes in Computer Science, vol. 293, Springer-Verlag, pp. 156–166, 1987.
- [6] F. Boudot. Efficient Proofs that a Committed Number Lies in an Interval. In proceedings *Eurocrypt 2000*, Lecture Notes in Computer Science, vol. 1807, Springer-Verlag, pp. 431–444, 2000.
- [7] R. Cramer and I. Damgård. Zero-Knowledge Proofs for Finite Field Arithmetic, or: Can Zero-Knowledge be for Free? In proceedings *Crypto '98*, Lecture Notes in Computer Science, vol. 1462, Springer-Verlag, pp. 424–441, 1998.
- [8] T. ElGamal. A public key cryptosystem and a signature scheme based on the discrete logarithm. *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.
- [9] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In proceedings *Crypto '86*, Lecture Notes in Computer Science, vol. 263, Springer-Verlag, pp. 186–194, 1986.
- [10] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal of Computing*, vol. 18, pp. 186–208, 1989.
- [11] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, vol. 38, pp. 691–729, 1991.

- [12] W. Mao. Guaranteed Correct Sharing of Integer Factorization with Off-line Shareholders. In proceedings *Public Key Cryptography '98*, Lecture Notes in Computer Science, vol. 1431, Springer-Verlag, pp. 60–71, 1998.
- [13] M. Stefik and A. Silverman. The Bit And the Pendulum: Balancing the Interests of Stakeholders in Digital Publishing. *The Computer Lawyer*, vol. 16, pp. 1–15, 1999.
- [14] ContentGuard: Rights Protection from Xerox. See <http://www.contentguard.com/>.
- [15] Intertrust Digital Rights Management. See <http://www.intertrust.com/>.
- [16] The Secure Digital Music Initiative. See <http://www.sdmi.org/>.